TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

HEIKKI SAARELMA

VERKKOSOVELLUSTEN TIETOTURVATESTAUS OSANA VERKKOHOTELLI-PALVELUA

Diplomityö

Tarkastaja: professori
Hannu Koivisto
Tarkastaja ja aihe hyväksytty
Automaatiotekniikan tiedekuntaneuvoston kokouksessa 7. toukokuuta 2014

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Automaatiotekniikan koulutusohjelma
SAARELMA, HEIKKI: VERKKOSOVELLUSTEN TIETOTURVATESTAUS OSANA VERKKOHOTELLIPALVELUA
Diplomityö, 65 sivua, 2 liitesivua
Syyskuu 2014
Pääaine: Automaatio- ja informaatioverkot
Tarkastaja: professori Hannu Koivisto
Avainsanat: Verkkohotelli, Verkkosovellus, Tietoturva, XSS, Injektio, Haavoittuvuusskanneri, OWASP ZAP, Vega

Verkkohotellipalvelu on yrityksille ja yksityisille asiakkaille tarjottava palvelu, jonka avulla voidaan ylläpitää verkkosivuja ja verkkosovelluksia. Verkkohotellipalvelu tarkoittaa yksinkertaisesti palvelin ja verkkokapasiteetin vuokraamista. Palveluiden suosio on kasvanut nopeasti ja samalla ennen palveluntarjoajiin kohdistuneiden hyökkäyksien painopiste on siirtynyt verkkosovelluksista etsittäviin haavoittuvuuksiin. Hyökkääjien ei ole enää kannattavaa luoda uusia vihamielisiä sivustoja, vaan vihamielisen ohjelmakoodin ujuttaminen jo tunnetuille hyvämaineisille sivuille on todettu tehokkaammaksi ratkaisuksi. Tällainen hyökkäys tarvitsee onnistuakseen verkkosivuilla sijaitsevan haavoittuvuuden ja haavoittuvuuksien etsiminen on yleisesti automatisoitu haavoittuvuusskannereilla. Työn yksi tavoite on tutustua tähän prosessiin ja pohtia miten palveluntarjoaja voi parantaa palveluaan ja asiakkaiden tietoturvaa.

Jotta palveluntarjoaja voi toimia tällaisessa ympäristössä, täytyy palveluntarjoajan ja asiakkaan vastuut olla etukäteen määritelty. Vastuiden määrittelyä varten täytyy ymmärtää verkkosovellusten rakenteita ja tietoturvaan liittyviä termejä. Lisäksi työssä tutustutaan yleisimpiin sovelluksia koskeviin uhkiin, kuten XSS-hyökkäyksiin, injektioihin ja tiedon vuotamiseen. Tämän tiedon pohjalta testataan kuusi dHosting-verkkohotellipalvelussa toimivaa verkkosovellusta. Testaus perustuu sovellusten tunnistamiseen ja kahden haavoittuvuusskannerin testausraportteihin. Tulosten pohjalta pohditaan löytyneitä haavoittuvuuksia, palvelun yleistä tietoturvan tilaa ja onko palveluntarjoajalla mahdollisuuksia parantaa omaa toimintaansa.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Automation Technology
SAARELMA, HEIKKI: Web Application Security Testing in Web Hosting Service
Master of Science Thesis, 65 pages, 2 Appendix pages
September 2014
Major: Automation and Information Networks
Examiner: Professor Hannu Koivisto
Keywords: Webhotel, Web Application, Information Security, Cross Site Scripting, Injections, Vulnerability Scanner, OWASP ZAP, Vega

Web hosting services are commonly used to upkeep websites and web applications. It simply means renting server and network capacity. During the years popularity of web have rocketed up and the focus of attacking service providers is slowly shifting to scanning web applications for vulnerabilities and misconfigurations. Also the attacking scheme have changed from setting up scam sites to plant hostile code to already reputable and popular sites. These attacks needs vulnerabilities to success. Finding these vulnerabilities is possible with automatized scanning tools and suitable software. One aspect of this thesis is to understand this process and to consider how the hosting provider can improve the quality of provided services and help customer with security issues.

In order to operate in this kind of environment must responsibilities between customers and providers be clear. Service provider should also have a general view about web applications and what is included to term information security. In addition to this we will also get familiar with most common threats and vulnerabilities like Cross-Site Scripting, injections and information leakages. Based on this information we will test six different web applications. Tests will be performed with two vulnerability scanners and two programs that are used to identify different applications and content management systems. Finally we will take a look to achieved test results and think about possibilities to improve the quality of webhosting service.

ALKUSANAT

Vaikka opiskeluideni pääaine oli automaatio- ja informaatioverkot, voisin nähdä opintojeni lopulta liittyneet pääasiassa internettiin ja sen rakenteeseen. Internet on maailman tehokkain työkalu tiedon jakeluun ja esittämiseen, mutta samalla se on tehokas kanava väärinkäytölle ja rikoksille. Huolenaiheita ovat internetin kaatuminen, terrorismi, kansalaisten tarkkailu, tietovuodot, sosiaalisen median aiheuttama riippuvuus, sekä tietenkin unohtuneet salasanat. Tähän kun yhdistetään kansainvälisesti vaihteleva lainsäädäntö ja internetin ei niin johdonmukainen kehitys, ollaan vähintäänkin sekavassa tilanteessa. Tietoturva ja siihen liittyvät ongelmat ovat saaneet huomattavasti lisää näkyvyyttä viime vuosina ja ne ovat listattu yhdeksi tulevaisuuden vakavimmista uhkakuvista. Diplomityön tekeminen avarsi ainakin omaa ymmärrystä aiheesta, ja muutti mielipiteeni kuinka laaja kokonaisuus kyseessä oikeasti on. Tietoturva ei ole konkreettinen asia joka voidaan korjata kuntoon, vaan se täytyy ennemmin nähdä prosessina tai mielentilana. Toivottavasti lukija löytää samoja ajatuksia tämän työn sivuilta.

Kiitokset työn valmistumisesta menee ohjaajalle Petteri Grönroosille, joka mahdollisti tämän projektin toteutumisen. Samoin työn tarkastajalle, professori Hannu Koivistolle, joka antoi työlle järkevät suuntaviivat. Suurin kiitos kuuluu kuitenkin avopuolisolleni Maiju Hyytiäiselle, joka jaksoi tsempata työn ääreen myös iltaisin, kun työpäivän seitsemän ja puoli tuntia täyttyivät muista tehtävistä.

Heikki Saarelma
Tampereella 19.9.2014

SISÄLLYS

Termit ja niiden määritelmät

Apache	Avoimeen lähdekoodiin perustuva verkkopalvelinohjelmisto
Domain Name System	DNS on järjestelmä, joka vertaa verkkotunnuksia ja IP-osoitteita sekä muuntaa niitä päikseen
Haavoittuvuus	Heikkous turvallisuusjärjestelmässä joka mahdollistaa järjestelmän väärinkäytön
Haavoittuvuusskanneri	Ohjelma joka etsii haavoittuvuuksia sovelluksista
HTML	Hypertext Markup Languge on yleisesti käytetty kieli verkkosivujen esittämiseen
HTTP	Hypertext Transfer Protocol on tilaton ja tekstipohjainen sovellustason hypertekstin siirtoprotokolla
HTTPS	SSL-tekniikalla salattu HTTP-liikenne
Hyökkäys	Ihminen tai järjestelmä joka käyttää haavoittuvuutta hyväkseen ei tarkoitetulla tavalla
IDS	Intrusion Detection Systems tarkoittaa tunkeutumisen tunnistusjärjestelmä
IIS	Internet Information Services on Windows palvelimelle asennettava verkkosivujen esittämistä varten tarvittava verkkopalvelinohjelma
IP-osoite	IP-osoite on numeromuotoinen esitys verkkosivun osoitteesta
IPS	Intrusion Prevention Systemson tarkoittaa tunkeutumisen estojärjestelmää
JavaScript	Yleisin selainpään ohjelmointikieli
Istunnonhallinta	Kaikki mitä tapahtuu siitä hetkestä, kun käyttäjä kirjautuu palveluun, aina siihen asti kun hän kirjautuu ulos palvelusta
Istuntotunniste	Tunniste jonka avulla eri istunnot tunnistetaan

REST	Representative State Transfer on verkkosovelluksissa käytetty hajautettu ohjelmistoarkkitehtuuri
Selain	Ohjelma verkkosivujen esittämiseen
Sisällönhallintajärjestelmä	Tietojärjestelmä verkkosovellusten luomiseen ja ylläpitoon
SOAP	Simple Object Access Protocol on XML-pohjainen verkkopalveluiden välinen viestintäprotokolla
SSL/TLS	Security Sockets Layer/ Transport Security Layer ovat viestiliikenteen salaamiseen käytettyjä protokollia
Tietoturvapolitiikka	Asiakirja jossa käsitellään tietoturvan säännöt, tavoitteet ja vaatimukset
Uhka	Mahdollinen tilanne tai tapahtuma joka voi aiheuttaa ei toivotun tilan
Vastatoimi	Järjestelmät, laitteet, toimet ja tekniikat joilla varaudutaan uhkiin
Verkkohotelli	(Engl. Web Hotel, Virtual Host, VHost) Verkkohotellipalvelin tarjoajalta vuokrattava virtuaalinen palvelinresurssi
Verkkopalvelu	Ohjelmistojärjestelmä, joka tukee yhteensopivien tietokoneiden välistä keskustelua verkon yli
Verkkopalvelin	HTTP-kielellä resursseja jakava tietokone
Verkkotunnus	IP-tunnukset selkokielinen nimi
Verkkosivu	Internetissä julkaistu HTML-dokumentti
Verkkosovellus	Verkkosivu joka generoi sisältönsä käyttäjän tai käyttäjän syötteiden perusteella
Välittäjäpalvelin	Palvelin joka välittää saapuneet pyynnöt toiselle palvelimelle
W3C	Internet-standardeja suunnitteleva yritysten ja yhteisöjen liitto

WSDL	Web Services Description Language on verkkopalveluiden kuvaamiseen käytetty XML-pohjainen kieli
XHR	XMLHttpRequest on palvelimen ja selaimen välinen rajapinta, joka mahdollistaa asynkronisen tiedonsiirron

1 JOHDANTO

Tämä diplomityö on tehty Decens Oy:lle, joka on Tampereella vuonna 2008 perustettu tietoteknisiä palveluja tarjoava yritys. Yrityksellä on konesali Tampereen alueella ja se täyttää viestintäviraston ohjeistuksen hyvästä konesalisuunnittelusta. Yritys on laajentanut toimintaansa verkkohotellipalveluiden tarjoamiseen vuonna 2013. Verkkohotellipalvelu toteutetaan dHosting-tuotemerkin alla ja sen yhteydessä työskentelee kehitysvaiheessa suoraan tai osallisesti 10 henkilöä.

Diplomityön tavoite on tutustua verkkosovellusten tietoturvaan, ymmärtää yleisimpiä uhkia ja luoda alalla toimintaan tarvittava ymmärrys. Samalla selvennetään asiakkaan ja palveluntarjoajan vastuualueita, parannetaan asiakkaiden tietoturvatietoisuutta ja pohditaan mahdollisuuksia tuotteistaa työn yhteydessä löytyviä parannusehdotuksia. Työn alkuperäinen tavoite oli testata verkkohotellipalveluun liittyvien komponenttien tietoturvan tasoa. Työn aikana todettiin, että verkkohotellipalvelun toteutus tehdään kolmannen osapuolen ohjelmistoilla, joiden asennus ja päivitys suoritetaan toimittajan ohjeiden mukaisesti. Näitä tutkimalla työssä ei todennäköisesti olisi saavutettu mielekkäitä tuloksia. Toinen vaihtoehto oli tutkia palvelun tuottamiseen käytettävien palvelimien asetuksia, mutta tätäkään ei nähty tarpeelliseksi. Mielenkiintoisin, ja todennäköisesti eniten tukea tarvitseva alue, todettiin olevan asiakkaiden verkkosivut ja -sovellukset. Ratkaistava kysymys päätettiin asettaa: miten verkkohotellipalveluntarjoaja voi auttaa asiakkaitaan tietoturvan parantamisessa? Testausvaiheessa kokeillaan, voiko palveluntarjoaja asettua hyökkääjän asemaan ja havaita mahdolliset haavoittuvuudet, ennen kuin ne johtavat mahdollisesti oikeaan hyökkäykseen. Lisäksi oletamme, että yleinen tietoturvan taso paranee, kun palveluntarjoaja ymmärtää mitä palvelun sisällä tapahtuu.

Verkkosivujen rakenne on kehittynyt staattisista HTML-dokumenteista dynaamisiksi verkkosovelluksiksi. Ensimmäisissä verkkosovelluksissa esiintynyt asiakas-palvelin-malli on poistumassa, kun verkkosovellukset ja selainohjelmat jakavat suoritettavia tehtäviä keskenään. Samalla painopiste palvelun rakenteisiin kohdistuvista hyökkäyksistä ja vihamielisistä verkkosivuista on siirtymässä verkkosovelluksista haavoittuvuuksia etsiviin automatisoituihin skannereihin. Laadukkaat sisällönjulkaisujärjestelmät ja viitekehykset ovat vähentäneet kriittisten haavoittuvuuksien kokonaismäärää, mutta löytyneet haavoittuvuudet ovat vakavampia ja soveltuvat useampaan kohteeseen. Verkkosovellukset ja niiden toteuttamiseen tarvittavat palvelinresurssit ovat vahvasti sidoksissa toisiinsa. Mahdollisen tietoturvahyökkäyksen tapahtuessa pohditaan, onko vastuu verkkohotellipalveluntarjoajalla, sovelluksen ylläpitäjällä vai loppukäyttäjällä. Palveluntarjoaja vastaa siitä, että palvelun toteuttamiseen käytetyt ohjelmistot ovat ajan tasalla ja verkkopalvelimien asetukset vastaavat suositeltuja käytäntöjä. Ellei asiakkaiden palveluun asentamat

sovellukset sisällä koko palvelun toimintaa häiritseviä komponentteja tai niiden sisältö riko Suomen lakia, on sovellusten tietoturva asiakkaan vastuulla.

Työn ensimmäinen osa koostuu teoreettisesta osuudesta. Luvussa kaksi käsitellään verkkosovellusten ja internetin yleisimpi tekniikoita, sekä miten verkkohotellipalvelu toteutetaan. Luvussa kolme tutustutaan termeihin tieto ja tietoturvallisuus. Luku neljä yhdistää kaksi aikaisempaa lukua ja käsittelee niiden yhteneviä ominaisuuksia juuri verkkohotellipalvelun kannalta. Lisäksi luvussa käsitellään yleisimmät verkkosovelluksia koskevat uhat ja miten niitä vastaan voidaan suojautua. Työn kokeellinen osuus alkaa luvusta viisi, jossa esitellään verkkohotellipalvelun asiakkaiden sovellusten tietoturvan testaamista. Luvussa viisi esitetyt testin suoritetaan testaukseen valituille kohdesovelluksille ja tulokset esitetään luvussa kuusi. Luku seitsemän sisältää testeissä esiin tulleiden asioiden pohdintaa, sekä niiden pohjalta syntyneet kehitysehdotukset. Työssä on rajattu tietoturvasyiden takia pois palvelun tuottamiseen tarvittavien laitteiden tai ohjelmien nimiä, joten työtä ei voi käyttää ohjeena verkkohotellipalvelun perustamiseen tai ylläpitoon.

2 VERKKOHOTELLIPALVELU JA VERKKO-SOVELLUKSET

Tässä luvussa käsitellään mitä tarkoittavat verkkohotellipalvelu ja verkkosovellus, miten ne toteutetaan ja mihin niittä käytetään. Webhotellivertailu.fi-sivuston vertailusta selviää, että Suomessa on yli 40 eri verkkohotellipalveluntarjoajaa. Listaukseen ovat päässeet vain suurimmat palveluntarjoajat ja pienien yritysten ja yksityisten palveluiden määrää on vaikea arvioida. Edullisimmat ratkaisut ovat asiakkaille ilmaisia ja kalleimmat yksityisillä palvelimilla toteutettavat ratkaisut maksavat tuhansia euroja kuukaudessa. Palveluntarjoajat vaihtelevat yhden henkilön pienyrityksistä monikansallisiin suuryrityksiin. Verkkohotelli tarkoittaa palvelinkapasiteetin vuokraamista, joka koostuu palvelimesta ja internetyhteydestä. Verkkohotellipalvelulle ei ole tarkkaa määritystä ja sen sisältö vaihtelee palveluntarjoajan mukaan. Verkkohotellipalveluun voi kuulua verkkotunnuksien ylläpito, sähköpostipalvelut, tietokantoja ja niiden hallintajärjestelmiä, tietojen varmuuskopiointi, esiasennettavia ohjelmia ja verkkohotellin hallintapaneeli. Verkkohotelleja käyttävät yritykset, yhdistykset, sekä yksityiset henkilöt. Palvelun sisältö vaihtelee automaattisesta kotisivukoneesta pelkän fyysisen raudan vuokraamiseen ja yleisin verkkohotellin käyttötarkoitus on verkkosivujen ylläpito [1].

Verkkohotelli on erinomainen esimerkki palveluliiketoiminnasta. Tarja Huovisen mukaan palvelu on aineeton suoritus, joka tuotetaan sekä kulutetaan samanaikaisesti, eikä sitä voi omistaa. Aineeton tuote joudutaan arvioimaan abstrakteilla mittareilla ja tämän takia palvelulle on suoritettava jatkuvaa laadunvalvontaa. Palvelu on ainutkertainen, sitä ei voida varastoida, säilyttää tai palauttaa, eikä asiakkaalle jää konkreettista tai jälleenmyytävää tuotetta. Koska palvelut tuotetaan samaan aikaan kuin ne kulutetaan, niitä ei voida erottaa palvelun toimittajasta [2].

Sana verkkosivu tarkoittaa internetissä julkaistua sivua ja verkkosovellus on verkossa toimiva ohjelma, joka tarjoaa palvelua selainsovelluksen kautta. Verkkosivu voidaan luokitella verkkosovellukseksi, jos se sisältää dynaamista sisältöä tai sen sisältö muodostuu hakusanojen, käyttäjän henkilöllisyyden tai käyttäjän lisäämän aineiston perusteella. Jokainen verkkosovellus on myös verkkosivu. Määritelmä ei ole tarkka ja jopa yksittäinen informatiivinen verkkosivu voidaan käsittää verkkosovellukseksi. Yksinkertaisimmillaan kyse on yhdestä HTML-sivusta ja laajimmillaan miljoonia käyttäjiä yhdistävästä sosiaalisesta sovelluksesta [3]. Verkkosovellus ei tarkoita samaa kuin verkkopalvelu. W3C on WWW:n kehittäjän, ja internetin isänä pidetyn, Tim Barners-Leen vuonna 1994 perustama ja johtama yritysten ja yhteisöjen liitto. W3C on määritellyt vuonna 2004 verkkopalvelun olevan ohjelmistojärjestelmä, joka tukee yhteensopivien tietokoneiden

välistä keskustelua verkon yli. Verkkosivun ja verkkosovelluksen ero voidaan nähdä olevan sisällön rakenteessa ja verkkosovelluksen ja verkkopalvelun ero loppukäyttäjässä. Asialle ei ole tieteellistä määritelmään ja termejä käytetään yleisesti ristiin. Tässä työssä noudatetaan edellistä esitystä [4].

2.1 Internetin historia ja selainohjelma

Asioita ymmärretään niiden historian kautta. Michael Zalewski kuvaa kirjassa The Tangled Web miten internet on kehittynyt nykyiseen muotoonsa [5 s.9-13]. Yhden tulkinnan mukaan internetin historia katsotaan alkaneeksi vuonna 1945. Tällöin Vannevar Bush kehitti tekniikan luoda viitteitä eri mikrofilmien välille, jotka etäisesti muistuttivat hyperlinkkejä. 1960 IBM loi GML-kuvauskielen (engl. Generilized markup language), jonka avulla tekstiin luotiin koneluvun mahdollistavia elementtejä, kuten otsakkeita ja listoja. Tutkijat Tim Barners-Lee ja Dan Connolly kehittivät näiden tekniikoiden pohjalta selvästi yksinkertaistetun HTML-kuvauskielen. HTML-dokumenttien välittämiseen kehitettiin HTTP-protokolla ja heidän työnsä kulminoitui 90-luvun alussa WorldWideWeb-nimisen selainohjelman julkaisemiseen. Selain tarkoittaa ohjelmaa, jolla voidaan esittää verkkosivuja. WWW-selaimen pohjalta kehitettiin Mosaic-niminen selainohjelma, joka mahdollisti esimerkiksi kuvien esittämisen osana HTML-dokumentteja. Mosaic-selaimen pohjalta kehitettiin uusia selainohjelmia, kuten Netscape Navigator ja Internet Explorer. Selainten suosion kasvaessa markkinoille ilmestyi selaimia, kuten Opera ja Lynx, jotka perustuivat omaa arkkitehtuuriinsa.

Kilpailusta seurasi selainohjelmien nopea kehitys, jossa HTML-standardi ei pysynyt perässä. Eri selaimet esittivät verkkosivuja eri tavoin ja tämä johti verkkosivujen selainkohtaiseen versiointiin, sekä tietoturvan ja käytettävyyden heikkenemiseen. W3C yritti saada standardin ajan tasalle keskeneräisinä julkaistuilla HTML 2.0 ja 3.2-versioilla, sekä julkaisemalla projekteja kuten CSS (Cascading Style Sheets). Samalla muut standardointijärjestöt, kuten ISO, IETF ja ECMA, yrittivät julkaista omia määrittelyjään eri protokollista. Näistä tärkeimpiä olivat HTTP-protokollan päivittäminen, selaimiin upotettava ohjelmointikieli JavaScript, sekä istuntotunnisteiden, kuten keksien, määrittely. Koska selainohjelmien kehitys oli yritysten vastuulla, osa määrittelyistä johti käyttämättömiin tai selainten kanssa ei yhteensopiviin standardeihin. Lopulta kilpailu rauhoittui 2000-luvun alussa, kun Internet Explorer-selain saavutti 80 prosentin markkinaosuuden ja muut valmistajat seurasivat suurimmaksi osaksi sen määrittelyjä. Tämän johdosta W3C ehti saada kehityksen kiinni ja julkaisi HTML 4 ja XHTML-standardit, teki merkittäviä muutoksia JavaScriptiin, dokumenttioliomalliin, sekä päivitti CSS-tyylikielen ajan tasalle. Päivityksistä huolimatta eri selainten tapa esittää verkkosivuja jatkui kirjavana. Kaikkia määrittelyjä ei otettu käyttöön, tai niistä tehtiin omia versioita.

Seuraava merkittävä vaihe selainten ja verkkostandardien osalta tapahtui vuonna 2004, kun Microsoft julkaisi XHR-palvelinrajapinnan (XMLHttpRequest). Se mahdollisti palvelimen ja selaimissa toimivien komentosarjakielien asynkronisen tiedonsiirron. Rajapinnan avulla komentosarjakielet pystyivät lähettämää verkkopalvelimille HTTP-

pyyntöjä ja lataamaan vastaukset osaksi selaimessa suoritettavaa ohjelmakoodia. Käytännössä verkkosivun elementin päivittäminen ei enää vaatinut koko sivun uudelleen lataamista. Samoihin aikoihin ongelmat eri verkkotekniikoiden tietoturvassa saivat enemmän huomiota ja Microsoftin hitaat toimenpiteet asioiden parantamiseksi avasi markkinoita uusille toimijoille. Käyttäjien ja tietoturva-asiantuntijoiden ylistämä Mozilla Firefox saavutti nopeasti 20 prosentin markkinaosuuden. Vaikka selainohjelma myöhemmin paljastui sisältävän yhtä lailla haavoittuvuuksia, sen avoimen lähdekoodin päälle rakennettu toimintamalli ja aktiivinen käyttäjäyhteisö auttoivat kehittäjiä korjaamaan esiintyneet ongelmat kilpailijoita nopeammin. XHR-rajapinnan menestys sai eri julkaisijat julkaisemaan kilpaa uusia, ei aina täysin valmiita, selainominaisuuksia, kuten web storage-selaintietokannan ja HttpOnly-keksit. Erimielisyydet kehityksen suunnasta johtivat W3C-yhteenliittymästä erkautuneen uuden WHATWG-järjestön (Web Hypertext Application Technology Working Group) syntymiseen. Kun W3C jatkoi XHTML 2.0-standardin kehitystä, WHATWG työskenteli HTML5-julkaisun parissa. HTML5 oli ensimmäinen kokonaisvaltainen ja tietoturvaan keskittyvä yhteenveto aikaisemmista tekniikoista, vaikka patenttikiistat Microsoftin kanssa hidastivat sen kehitystyötä. HTML-kieltä käydään tarkemmin läpi kappaleessa 2.2, mutta lyhyestä historiasta huomataan, internetin kehitys on ollut nopea, yritysvetoinen kilpajuoksu ilman yhteistä päämäärää.

Todennäköisesti vanhentuneet selaimet, esimerkiksi IE 6, pystyisivät esittämään suurimman osan nykypäivän verkkosivuista. Sivuja harvoin enää suunnitellaan vain yhdelle selaimelle, vaikka tietoturvamielessä vanhojen selaimien tuen lopettaminen olisi järkevää. Selaimen ydin on renderöintimoottori. IE-selaimen moottori on Trident, Safarilla ja Chromella on WebKit, Firefoxilla Gecko ja Operalla Presto. Renderöintimoottori tulkitsee palvelimen lähettämän HTML-tiedoston dokumenttioliomalliksi, suorittaa komentosarjakielen ohjelmakoodin, muodostaa verkkosivujen ulkoasun, sekä vastaa lopulta verkkosivun tietoturvasta [6 s.xvi]. Yhden lähteen politiikka on yksi ensimmäisistä selainten tietoturvaominaisuuksista. Se määrittää, että selaimessa suoritettava verkkosivun ohjelmakoodi ei voi keskustella toisella sivulla sijaitsevan ohjelmakoodin kanssa. Verkkosivu voi ladata resursseja toisesta lähteestä, mutta ei näe toisen sivun dokumenttiomallia, ei voi asettaa keksejä, injektoida ohjelmakoodia tai pysty lukemaan toiselle sivulle lähetettyjen pyyntöjen vastauksia. Lähde tarkoittaa, että suoritettavalla koodilla on sama protokolla, portti ja isäntä. Ominaisuus on rajoittanut verkkosovellusten toimintaa ja sen kiertämiseksi HTML5-versioon on kehitetty CORS-kättely (Cross Origin Resource Sharing). CORS-kättely alkaa, kun verkkosovellus lähettää selaimelle pyynnön XHR-rajapinnan kautta, jossa se pyytää selainta tiedustelemaan toiselta verkkosovellukselta, suostuuko se resurssien jakamiseen. Kohdesovellus voi suostua kättelyyn ja palauttaa HTTP-viestin otsakkeessa tiedon siitä, mitkä sen resursseista ovat käytössä, minkä tyyppiset HTTP-pyynnöt ovat mahdollisia, kauan CORS-kättely on voimassa ja voiko toinen sovellus käyttää sen istuntotunnisteita. Kun kättely on suoritettu, selain keventää yhden lähteen politiikkaansa, ja verkkosovellukset voivat aloittaa resurssien vaihtamisen. CORS-kättely on tapa ohittaa tietoturvaominaisuus, mutta oikein käytettynä sen avulla voidaan

määritellä tarkemmin eri verkkosovellusten resurssien käyttöoikeudet, kuin pelkällä yhden lähteen politiikalla [6 s.3-6]. Resurssien jakaminen on johtanut myös uusien hyökkäysten syntymiseen. Tästä esimerkkinä luvussa 4 esiteltävä XSRF-hyökkäys, joka perustuu vihamielisien komentojen toimittamiseen selaimen kautta.

CORS-menettely ja yhden lähteen politiikka ovat yksi esimerkki, mutta selaimella on tärkeä tehtävä internetin tietoturvassa. Selain on usein myös ainoa ohjelma, joka vaaditaan verkkosovellukseen kohdistuvan hyökkäyksen suorittamiseen. Hyökkäys voi perusta monimutkaiseen rekisterien ylivuotojen pitkäaikaiseen keräämiseen ja tulkintaan, tai yksinkertaisesti pelkän URI-tunnisteen muuttaminen. Toinen tärkeä työkalu hyökkäysten muodostamiseen on ohjelma, jolla voidaan generoida ja lähettää HTTP-pyyntöjä. Netcat on yksi ensimmäisistä tällaisista ohjelmista ja se toimi avaamalla verkkosoketteja, lähettämään niihin pyyntöjä ja lukemalla vastauksen. Soketti on abstrakti käsite ja tarkoittaa verkkokommunikaation päätepistettä. Esimerkiksi IP-osoite ja verkkoportti määrittävät tällaisen päätepisteen. Tässä työssä käytetyt haavoittuvuusskannerit pystyvät tallentamaan käyttäjän selaimen liikenteen muistiinsa ja tarjoavat graafisen työkalun palvelinpyyntöjen muokkaamiseen [6 s.xvi].

2.2 HTML ja HTTP

HTML on yleisin tapa esittää verkkosivuja ja sen viimeisin versio on HTML5. Kilpailevan XHTML 2.0-standardin kehitys lopetettiin vuonna 2009. Ensimmäinen versio HTML5-standardista julkaistiin vuonna 2008 ja W3C on ilmoittanut, että sen tavoite on saada lopullinen versio julkaistua vuoden 2014 loppuun mennessä. Selain tietää tulkitsevansa HTML5-standardin mukaista verkkosivua HTML-dokumentin alussa olevasta <!doctype html>-esittelystä. Näkyviä muutoksia aikaisempiin HTML-versioihin ovat uudet elementit, sekä UTF-8 on määrittely oletuksena käytettäväksi merkistöstandardiksi. Lisäksi HTML5-standardin kehityksessä on mietitty aikaisempia versioita enemmän tietoturvaominaisuuksia. Epätieteellinen vertailu paljastaa, että sana ”turvallisuus” esiintyy HTML4-standardissa 14 kertaa ja HTML5-standardissa 73 kertaa. Selkein tietoturvaa parantava ominaisuus on selaimille asetetut tiukentuneet säännöt verkkosivujen esittämisestä, joiden on tarkoitus vähentää selainkohtaista vaihtelua sivujen esityksessä ja helpottaa sovelluskehittäjien työtä [6 s.1-3].

Syntaksiltaan HTML on yksinkertaista. Verkkosivut esitetään hierarkkisena puumallina käyttäen elementtejä. Elementit erotetaan hakasuluilla rajatuilla alku- ja lopputunnisteilla, joista alkutunnisteessa voidaan asettaa tarkentavia parametreja. Yksinkertainen verkkosivu, jossa on otsikko, nimike ja linkki toiselle sivulle, voisi olla rakenteeltaan ohjelmaesimerkin 2.1 kaltainen.

```
<html>
  <head>
    <title>Hello world</title>
  </head>
  <body>
      <h1>Esimerkkisivu</h1>
      <a href="http://www.esimerkki.fi/">Linkin teksti!</a>
  </body>
</html>
```

Ohjelma 2.1. Esitys yksinkertaisesta HTML-sivusta joka sisältää otsikon ja linkin

Merkit, kuten et-merkki (&), hakasulut, sekä heitto ja –lainausmerkit, ovat varattu HTML-syntaksin käyttöön. Niiden käyttöä pitäisi välttää tai tarvittaessa ne suoritetaan pakomerkin avulla. Määrittelyyn on useita erikoistapauksia ja poikkeuksia, kuten elementtien nimet eivät ole merkkikokoriippuvaisia, lainausmerkkien käyttöä ei aina vaadita ja jotkin elementit, kuten kuvat, eivät tarvitse lopputunnistetta. Jos elementtiä ei osata tulkita oikein, jatkaa verkkoselain seuraavan elementin esittämisellä. Yleisesti HTML-elementit ovat puumallisesti toistensa sisällä. Poikkeuksen tähän tekevät esitettävästä HTML-kielestä poikkeavat erikoiselementit, kuten <style> ja <script>. Nämä elementit ovat merkkikokoriippuvaisia ja ne suoritetaan vasta kun vastaava lopputunniste on löydetty. Näiden elementtien sisälle sijoitettuja toisia elementtejä ei käännetä osaksi dokumenttioliomallia [6 s.72].

Eri selaimet tulkitsevat HTML-kielen virheitä ja poikkeuksia vaihtelevasti. Huonosti suunniteltu HTML-elementti voisi olla seuraavan kaltainen:

```
<i <b> > tekstiä </i>
```

Esimerkissä elementin parametrissa on käytetty hakasulkua. Useat selaimet ymmärtävät että kyseessä on i-elementti ja <b>-merkkijono on virheellinen parametriksi, joka jätetään tulkitsematta. Firefox-selaimen aikaisemmat versiot tästä poikkeavasti sulkivat i-elementin kohdatessaan hakasulun ja tulkitsivat, että kyseessä on uusi b-elementti. Vastaavia esimerkkejä on useita, kuten tilanne jossa esimerkiksi elementti unohdetaan sulkea. Osa aikaisemmista Opera- ja Firefox-selaimista tulkitsivat kääntäjän syntaksia takaperoisesti ja tulkita kyseessä olen b-elementti [5 s.76]. Vaihtelevat standardien tulkintatavat mahdollistavat erilaisia hyökkäyksiä. Esimerkiksi XSS-hyökkäyksessä JavaScript-ohjelmakoodia ujutetaan HTML-kielen sekaan käyttämällä hyväksi epätarkkaa määrittelyä, huonosti suunniteltuja syötekenttiä ja selainten välisiä tulkintaeroja. Verkossa on palveluja ja tietokoneelle voidaan ladata ohjelmia, jotka tulkitsevat verkkosivujen HTML-syntaksia, validoivat vastaako sivun rakenne haluttua standardia, sekä tunnistavat virheellisiä ja vaarallisiksi tunnettuja malleja. Käyttäjän kannalta paras keino parantaa tietoturvaa on käyttää nykyaikaista ja päivitettyä selainta.

HTTP tulee sanoista Hypertext Transfer Protocol ja se on tilaton ja tekstipohjainen sovellustason hypertekstin siirtoprotokolla. Sen ensimmäinen versio oli Tim Barnels-Leen vuonna 1991 julkaisema puolentoista sivun dokumentti. Ensimmäinen virallinen versio HTTP/1.0 on määritetty RFC 1945-esityksessä ja versio 1.1. RCF 2616-esityksessä kesäkuussa 1999. HTTP perustuu TCP/IP-protokollayhdistelmään. Jokainen HTTP-istunto aloitetaan muodostamalla TCP-yhteys oletuksena palvelimen verkkoporttiin 80. Yhteyden muodostamisen jälkeen, selain tekee resurssipyynnön palvelimelle ja palvelin palauttaa resurssin pyynnössä olevan URL-osoitteen avulla. URI on lyhenne sanoista Unified Resource Identifier ja se avulla nimetään verkkopalvelimella olevat resurssit. URL on lyhenne sanoista Unified Resource Locator. Sen avulla määritetään verkkoresurssin sijainti ja se sisältää URI:sta poiketen myös tiedon resurssin palvelumuodosta (scheme). Resurssi voi olla HTML-verkkosivu, kuva, videotiedosto, verkkopalvelu, yms. Yksinkertaisessa esimerkissä palvelin katkaisee TCP-yhteyden heti resurssin lähettämisen jälkeen. HTTP on tietoturvamielessä ongelmallinen protokolla. Se on tilaton, viestit ovat selkokielisiä ja syntaksin tulkitseminen ei vaadi erikoisosaamista. Tilattomuus tarkoittaa, että jokainen pyyntö käsitellään aina samalla tavalla riippumatta käyttäjästä tai aikaisemmista pyynnöistä. Tämän lisäksi protokolla toimii yleisesti tunnetun verkkoportin kautta ja verkkopalvelimien palomuurit ovat yleisesti asetettu sallimaan kaiken liikenteen tähän porttiin [7][8 s.8].

Alkuperäinen HTTP/0.9-protkolla sisälsi tekniikan vain resurssipyynnön lähettämisen. HTTP/1.1-standardissa viestiin lisätään otsakkeiksi kutsuttua metadataa. Otsakkeen nimi ja tietokenttä on erotettu kaksoispisteellä ja jokainen otsake on omalla rivillään. Ohjelmassa 2.2 on esitetty esimerkki HTTP/1.1-pyynnön rakenteesta.

```
POST /verkkosivu/palvelu.php HTTP/1.1
Host: www.esimerkki.fi
User-Agent: Esimerkkiselain/versionumero 1.0
Content-Length: 22
Referer: http://www.esimerkki.fi/index.html

PYYDÄN VERKKORESURSSIA
```

Ohjelma 2.2 HTTP-pyyntö esimerkki

Esimerkissä määritetään ensimmäisellä rivillä pyynnön tyyppi, kohde ja käytettävä protokolla. Host-otsake määrittää kohdepalvelimen osoitteen, User-agent-otsake käytössä olevaa selainohjelman ja Referer-otsake ilmoittaa miltä verkkosivulta pyyntö on tehty. Erilaisia otsakkeita on protokollasta riippuen useita kymmeniä. Otsakkeet loppuvat tyhjään riviin, jonka jälkeen tulee viestin hyötykuorma. Hyötykuorman pituus ilmoitetaan Content-Length-otsakkeella. Palvelimen mahdollinen vastaus tähän kyselyyn on esitetty ohjelmassa 2.3.

```
HTTP/1.1 200 OK
Server: Esimerkki-palvelin/1.0
Content-Type: text/plain
Connection: close

RESURSSIN SISÄLTÖ
```

Ohjelma 2.3 **Esimerkki HTTP-pyynnön vastauksesta**

Palvelin vastaa kertomalla käytetyn protokollan, protokollan numeroidun tilaviestin, sekä mahdollisesti tekstimuotoisen viestin. Viestiin voidaan lisätä otsikkokenttiä ja lopulta tyhjän rivin jälkeen lähetetään pyydetty resurssi. Samoin kuin HTML-dokumenttien tulkitsemisessa, eri selaimet tulkitsevat otsakkeita ja niissä esiintyviä virheitä toisistaan poikkeavasti. Esimerkiksi saman otsakkeen esiintyessä kaksi kertaa, osa selaimista tulkitsee ensimmäisen ja osa jälkimmäisen olevan oikea otsake. Myös verkkopalvelinsovellukset käsittelevät pyyntöjä ja otsakkeita toisistaan poikkeavasti. Apache-verkkopalvelin tulkitsee ensimmäisen Host-otsakkeen olevan oikea, kun taas IIS-verkkopalvelin hylkää koko pyynnön, jossa kyseinen otsake esiintyy kahdesti [5 s.42-43,47].

HTTP ei ole ainoa verkkosovelluksissa käytetty protokolla. Se käsittelee ASCII-pohjaisia viestejä, mutta esimerkiksi binääristen viestien ja tiedostojen lähettämiseen on kehitetty MIME-laajennus (RFC2046 - Multipurpose Internet Mail Extensions). MIME-protokollaa käytetään lisäksi eri merkistöjä käyttävien viestien lähettämisessä. SMTP (Simple Mail Transform Protocol) on TCP-pohjainen protokolla, jota käytetään sähköpostipalvelimien kesken ja FTP-protokollaa (File Transfer Protocol) käytetään tiedostojen siirtämiseen koneiden välillä. HTTP-protokollalle on olemassa myös WebDAV-niminen laajennus (RFC2518 – Web Distributed Authoring and Versioning), jonka avulla verkkosivuja ja palvelinresursseja voidaan lataamisen lisäksi hallita ja päivittää [8 s.10].

2.3 Istunto ja tietoliikenteen salaaminen

Aikaisemmin on todettu, että HTTP on tilaton protokolla. Mikään ei liitä protokollan pyyntöjä toisiinsa, aseta pyyntöjä tiettyyn järjestykseen tai vaadi, että käyttäjän pyynnöt tulevat yhdestä IP-osoitteesta. Useat verkkosovellukset kuitenkin tarvitsevat tällaisia ominaisuuksia. Istunto on mekanismi, jolla verkkopalvelu ylläpitää yhteyttä käyttäjän ja palvelimen välillä ja istunnonhallinta käsittää kaiken mitä tapahtuu siitä hetkestä, kun käyttäjä kirjautuu palveluun, aina siihen asti kun hän kirjautuu ulos palvelusta. Verkkosovelluksissa tämä on pääasiassa toteutettu selaimeen ladattavilla istuntotunnisteilla (engl. session token). Yleisin istuntotunniste on eväste, mutta se voi olla esimerkiksi osa URI-tunnistetta tai piilotettu kenttä HTML-lomakkeessa [6 s.142]. Eväste-mekanismi perustuu palvelimen selaimelle lähettämään Set-Cookie-otsakkeeseen, jonka selain tallentaa ja lähettää takaisin HTTP-pyynnöissä Cookie-otsakkeena. Palvelimen lähettämään otsakkeeseen voidaan lisätä parametreja, kuten kuinka kauan eväste on voimassa, koskeeko

evästä vain yhtä palvelua vai koko verkko-osoitetta, sekä tietoturvaominaisuuksia. Esimerkiksi HttpOnly-ominaisuus rajoittaa JavaScriptin pääsyä evästeeseen [5 s.60-61].

Tarkasti ottaen eväste tunnistaa selaimen, ei käyttäjää, mutta näiden voidaan olettaa olevan sama asia. Kun käyttäjä on todennettu, verkkopalvelin luottaa istuntotunnisteeseen ja tämän takia tunnisteen rakenne ei saa olla ennalta-arvattava. Istuntotunnisteita vastaan voidaan hyökätä useilla eri tavoilla. XSS-hyökkäyksessä JavaScriptillä voidaan vaikuttaa keksin ominaisuuksiin ja CSRF-hyökkäys perustuu eri verkkosovellusten istuntotunnisteiden hyväksikäyttöön. Verkkosovellukset voivat tallentaa istuntotunnisteita tietokantaan säästääkseen järjestelmän muistia ja tällöin ne ovat vaarassa tietokantainjektioille. Jos HTTP-yhteys ei ole salattu, istunnontunnisteen välittyvät selkokielisinä ja ovat helposti kaapattavissa paketteja nuuskimalla [6 s.143-144]. Kaikki uhkat koskien istuntotunnistetta eivät aina vaadi hyökkäävää osapuolta. Esimerkki huonosti suunnitellusta istuntotunnisteesta on esitetty ohjelmassa 2.4, jossa istuntotunniste on osana URI-tunnistetta.

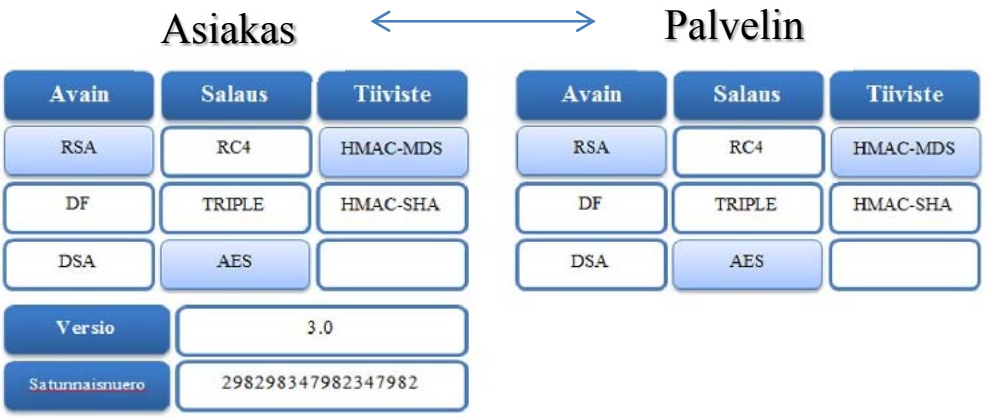
```
http://lentoyhtio.fi/myynti/tuote;sessio-
nid=2P0OC2JSNDLPSKHCJUN2JV?kohde=Tampere
```

Ohjelma 2.4 **Esimerkki huonosti suunnitellusta istuntotunnisteesta**

Esimerkissä käyttäjä katsoo lentoyhtio.fi -verkkosivuilta lennon ja lähettää tästä linkin toiselle käyttäjälle. Koska tunniste on osa verkko-osoitetta, niin toinen käyttäjä voi vahingossa käyttää tilauksessaan alkuperäisen käyttäjän istuntotietoja tai pahimmassa tapauksessa vaikka luottokorttitietoja [9 s.8].

Jokainen verkkosolmu selaimen ja palvelimen välissä voi lukea HTTP-liikennettä. Yleisin tapa suojata HTTP-liikenne on tunneloida verkkoliikenne SSL-protokollan läpi ja tällöin puhutaan HTTPS-protokollasta. Lyhenne tulee sanoista Hypertext Transfer Protocol Secure. SSL on lyhenne sanoista Security Sockets Layer ja sen tehtävä on salata siirtokerroksella välitettävä tieto. SSL-protokollan uusin versio on nimeltään Transport Security Layer ja siitä käytetään lyhennettä TLS. TSL-protokollaan uusin version on vuonna 2008 julkaistu TSL 2.0, mutta yleisesti molemmista protokollista käytetään yhteistä termiä SSL. Alkujaan SSL oli Netscape Navigator-selaimen yhteyteen kehitetty lisäosa, mutta myöhemmin sen kehitys on luovutettu IETF standardointiorganisaatiolle. SSL ei rajoitu ainoastaan HTTPS-yhteyksiin, vaan sitä käytetään lisäksi myös sähköpostiprotokollien suojaamiseen [8 s.9][10][11].

HTTPS-pyyntö tunnistetaan URI-osoitteesta, joka alkaa https://-merkkijonolla. Oletuksena HTTPS-protokolla ottaa tällöin yhteyden verkkopalvelimen porttiin 443. HTTPS-yhteys alkaa kättelyvaiheelle, joka on esitetty kuvassa 2.1.



Kuva 2.1 **SSL-kättelyvaihe ja salausmetodin valitseminen**

Kuvassa asiakas lähettää kättelyviestin palvelimelle. Viesti sisältää asiakaskoneella käytössä olevat avaintenvaihtometodit, salaustavat sekä tiivisteen luomiseen tarvittavat tekniikat. Tiivistettä käytetään tiedon eheyden varmistamisessa. Lisäksi viesti sisältää tiedon käytettävästä SSL-versiosta, sekä salausavaimen laskemiseen tarvittavan satunnaisnumeron. Avaimet generoidaan yksilöllisesti jokaiselle yhteydelle. Palvelin vastaa kättelyviestiin ja ilmoittaa mitä avainta, salausta ja tiivistettä käytetään [12].

SSL-protokollaa käytetään lisäksi tunnistautumiseen. Palveluntarjoaja hankkii SSL-sertifikaatin niitä tarjoavalta yritykseltä (engl. Certificate Authority, CA). Tunnettuja sertifikaatin tarjoajia ovat esimerkiksi Symatec Group ja Comodo[13]. Sertifikaatin saamiseksi palveluntarjoaja lähettää Sertifikaatin myöntäjälle tietoja palvelimesta, kuka sertifikaattia tilaava yritys on ja missä se sijaitsee. Sertifikaatin myöntäjä tarkistaa nämä tiedot, luo uuden sertifikaatin, allekirjoittaa sen ja antaa sille voimassaoloajan. Palveluntarjoaja tallentaa myönnetyn sertifikaatin palvelimelleen. Selaimissa on sisäänrakennettu tiedot sertifikaattientarjoajista, ja vertaamalla tätä tietoa palvelimella olevaan sertifikaattiin, selain voi päätellä kuka palvelimen omistaa ja onko sivusto luotettava [11][12].

SSL-protokolla ja niitä tarjoavat yritykset voivat myös joutua tietoturvahyökkäysten kohteeksi. Vuonna 2011 Iranilaiset krakkerit murtautuivat Comodo Group:in varmenteita myöntävän kumppanin palvelimelle ja saivat varmenteita haltuunsa. Myönnettyjen varmenteiden peruminen käytöstä estäisi niitä ostaneiden asiakkaiden sivujen toiminnan. Myös vuonna 2014 paljon huomiota herättänyt Heartbleed-haavoittuvuus perustui avoimen OpenSSL-sertifikaatin salauskirjaston ohjelmointivirheeseen.

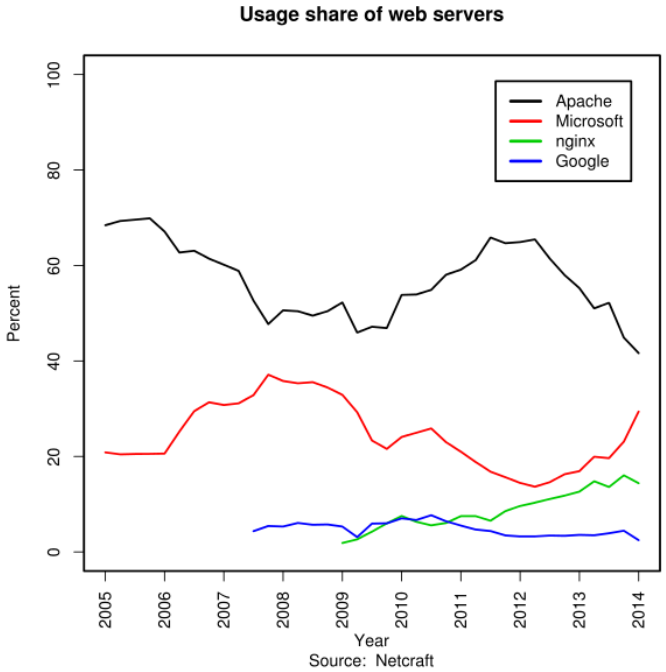
Tietoliikenteen salaaminen ei kuitenkaan suojaa hyökkäyksiltä jotka tapahtuvat joko palvelimella tai selaimessa. Esimerkiksi XSS-hyökkäys suoritetaan palvelimella toimivan verkkosovelluksen sisällä ja vaikka yhteys resurssiin olisi salattu, ei se ota kantaa onko esitetty sisältö laillista vai laitonta [39]. Osa XSS-hyökkäyksen vaarallisuudesta perustuukin juuri hyvämaineisen sivun uskottavuuteen. SSL-yhteys ei suojaa myöskään CSRF-hyökkäykseltä. Jos käyttäjän selain saadaan lähettämään laiton pyyntö käyttäjän tietämättä, hoitaa SSL-protokolla vain sen salaamisen ja toimittamisen. Koska pyyntö vaikuttaa tulevan alkuperäiseltä käyttäjältä, palvelin ei pysty tekemään eroa vihamielisen

resurssipyynnön ja aidon käyttäjän lähettämästä pyynnön välillä [40]. Yhteyden salaaminen vaikeuttaa näihin hyökkäyksiin tarvittavien istuntotunnisteiden kaappaamista ja näin epäsuorasti lisää käyttäjien tietoturvallisuutta. Tietoliikenteen salaamisen lisäksi palvelimelle ja käyttäjän koneelle asennetut tietoturvaohjelmistot, jotka oppivat ja tunnistavat eri hyökkäysmalleja, ovat parempi tapa suojautua XSS- ja CSRF-hyökkäyksiä vastaan.

2.4 Verkkopalvelin

Verkkopalvelimen ovat tietokoneita, joilla pidetään verkkosivuja ja -sovelluksia. Verkkopalvelimella on oma IP-osoite ja mahdollisesti IP-osoitteeseen liitetty verkko-osoite. Normaali tietokone voidaan muuttaa verkkopalvelimeksi, kun siihen asennetaan verkkopalvelinohjelmisto ja se liitetään internetiin. Tämän jälkeen verkkopalvelin toimii palveluna, johon asiakasohjelma lähettää pyynnön resurssista URL-osoitteen perusteella. Palvelin käsittelee pyynnön, tarkistaa että resurssi on olemassa, ja palauttaa sen asiakasohjelmalle. W3Techs.com-sivusto on vertaillut verkkopalvelinsovelluksia toteutettujen asennuksien määrän mukaan [13]. Tutkimus osoittaa, että vuonna 2014 yli 60 prosenttia verkkopalvelimista käytti Apache-verkkopalvelinohjelmistoa. Toiseksi suosituin ohjelmisto on Nginx (20%) ja kolmanneksi IIS (14%). IIS tulee sanoista Internet Information Services ja se on Microsoftin kehittämä verkkopalvelinohjelmisto.

Toisessa tutkimuksessa englantilaisen Netcraft-nimisen yritys on vertaillut verkkopalveluiden suosiota loppukäyttäjämäärien mukaan [14]. Tutkimuksen tuloksia on esitetty kuvassa 2.2.



Kuva 2.2 Verkkopalvelimien suosion kehitys käyttäjämäärällä mitattuna [14]

Kuvan trendiviivoja seuraamalla huomataan, että Microsoftin IIS-verkkopalvelinohjelmisto voi lähivuosina olla käyttäjämäärällä mitattuna suosituin verkkopalvelinsovellus.

dHosting-palvelu on toteutettu sekä Apache että IIS verkkopalvelinasennuksena, joista molempien päälle on asennettu oma versio Plesk-hallintapaneelisovelluksesta. IIS-verkkopalvelimelle on asennettu Parallels Plesk for Windows-sovellus, joka toimii käyttöliittymänä verkkohotellien luomiseen ja ylläpitoon. IIS-verkkopalvelimella verkkosivut voivat käyttää Microsoftin Access-tietokantoja sekä .Net-viitekehystä. Yleisesti Windows-pohjaista ratkaisua suositellaan vain, jos sitä ehdottomasti vaaditaan verkkosovelluksen toteuttamiseen. Parallels Plesk for Linux -asennus käyttää Apache-verkkopalvelinta verkkosivujen ylläpitoon. Apache-verkkopalvelinasennus voidaan täydentää välittäjäpalvelimena toimivana Nginx-laajennuksella. Laajennus tehostaa palvelimen saavutettavuutta suurilla käyttäjämäärillä, koska Nginx käsittelee Apachea tehokkaammin yhtäaikaisesti tulevia pyyntöjä ja tehostaa käyttäjäkohtaista muistinhallintaa [15 s.24-26][16].

2.5 Verkkosovellukset

Tässä kappaleessa käydään läpi yleinen verkkosovelluksen rakenne ja sen tärkeimmät komponentit. Verkkosovellus on ohjelma, jota voidaan käyttää verkkoselaimella tai erillisellä agenttiohjelmalla. Selain lähettää verkkopalvelimelle pyynnön, jonka palvelin ohjaa oikealle verkkosovellukselle suoritettavaksi. Verkkosovellus suorittaa pyynnölle tarvittavat toimenpiteet, palauttaa resurssin verkkopalvelimelle, joka lopulta palauttaa käyttäjälle selaimessa esitettävän HTML-sivun tai muun resurssin. Verkkosovelluksen rakenne on usein kerrostettu tietoturvan, paremman ylläpidon ja tehokkaamman palvelun takaamiseksi [17].

2.5.1 Verkkosovellusten historia

Jari-Pekka Vuotilaisen jakaa diplomityössään verkkosivujen historian kolmeen sukupolveen [18]. Ensimmäinen vaihe alkoi 1990-luvun alkupuolella ja tällöin sivut koostuivat staattisista HTML-tiedostoista. Toinen sukupolvi voidaan katsoa alkaneeksi 1995, kun JavaScript-ohjelmointikieli esiteltiin ja komponenttien, kuten Flash, QuickTime ja Shockwave, lisäämisen osaksi verkkosivuja tuli mahdolliseksi. Uudet tekniikat mahdollistivat animaatioiden ja liikkuvan kuvan liittämisen verkkosivuille ja verkkosovellukset alkoivat kehittyä multimediaesitysten suuntaan. Kolmas sukupolvi käsittää rikkaat verkkosovellukset (engl. Rich Internet Applications, RIA). Rikas verkkosovellus tarkoittaa työpöytäsovelluksen kaltaista ohjelmaa, jossa verkkosivut ja taustapalvelimet keskustelevat asynkronisesti. Verkkosivujen yksittäisiä elementtejä pystyttiin päivittämään ilman, että koko sivu täytyi ladata uudelleen. HTML5-standardin myötä lisäosien tarve vähentyi ja toimintoja on lisätty selainohjelmien ominaisuuksiin.

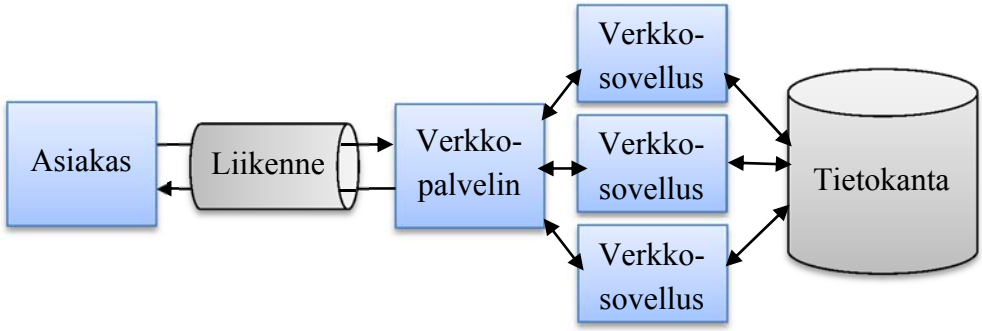
Vuonna 1999 keksittiin markkinointitermi Web 2.0, joka yleistyi kolmannen sukupolven myötä vuonna 2004. Se ei ole tekninen määrittely, vaan tapa jolla verkkosivuja ohjelmoidaan ja käytetään. Muutosta kuvaa verkkosivujen käytön siirtyminen ”vain luku”-tyylisistä sivustoista ”luku ja kirjoitus”-tyylisiksi sovelluksiksi. Verkkopalveluiden käyttäjistä alkoi muodostua yhteisöjä, jotka tuottivat itse sisältöä palveluihin. Myös verkkosovellukset alkoivat jakaa tietoa keskenään ja päivittämään itse itseään keräten sisältöä

toisista sovelluksista. Sovellusten keskeistä tiedonsiirtoa varten kehitettiin WSDL-kieli (Web Service Description Language) [64]. WSDL perustuu XML-kieleen ja sen avulla voidaan kuvata missä tietty verkkopalvelu sijaitsee ja mitä funktioita se tarjoaa. Sovellusten väliseen tiedonsiirtoon käytetään pääasiassa SOAP-protokollaa (Simple Object Access Protocol). SOAP-protokolla on myös XML-kieleen perustuva tietoliikenneprotokolla jonka mahdollistaa esimerkiksi HTTP-protokollan yli toteutettavat etäkutsut (RPC). Tämä tarkoittaa, että eri käyttöjärjestelmien päällä toimivat järjestelmät ja eri ohjelmointikielet pystyivät keskustelemaan keskenään.

Koska käyttäjillä ei ole tarvetta nähdä tai ymmärtää koneiden välistä keskustelua, kehitettiin tietokoneiden väliseen tiedonsiirtoon XML-pohjainen RDF-viitekehys (Resource Description Framework). RDF-viitekehyksen tarkoitus on määrittää ja standardisoita verkkoresursseja, niin että ne ovat yhteneviä keskenään. Yleisiä viitekehyksen määrittämiä kohteita ovat esimerkiksi tieto resurssin ylläpitäjästä, luontipäivä, milloin resurssia on viimeksi muokattua tai hakukoneille tarkoitettua metatietoa. RDF-viitekehys on osa W3C-järjestön semanttisen verkon visiota, jossa kaikilla verkkoresursseilla on tarkka määritelmä ja tietokoneet pystyvät tulkitsemaan resurssien sisältöä. Tästä käytetään joskus termiä Web 3.0. Eräiden määrittelyjen mukaan tällä tarkoitetaan semanttista tietoa ja tekoälyä hyödyntäviä verkkopalvelimia ja verkkosivujen suunnittelu hakukoneita ja -robotteja silmälläpitäen [19][64].

2.5.2 Verkkosovelluksen rakenne

Verkkosovellusten rakenne muistutti aluksi hyvin paljon asiakas-palvelin mallia. Asiakassovellus lähetti palvelimelle pyynnön ja jos se oli oikeamuotoinen, lähetti palvelin takaisin pyydetyn resurssin. Eräs mahdollinen verkkosovelluksen arkkitehtuuri on kuvattu kuvassa 2.3.



Kuva 2.3 Verkkosovelluksen rakenne

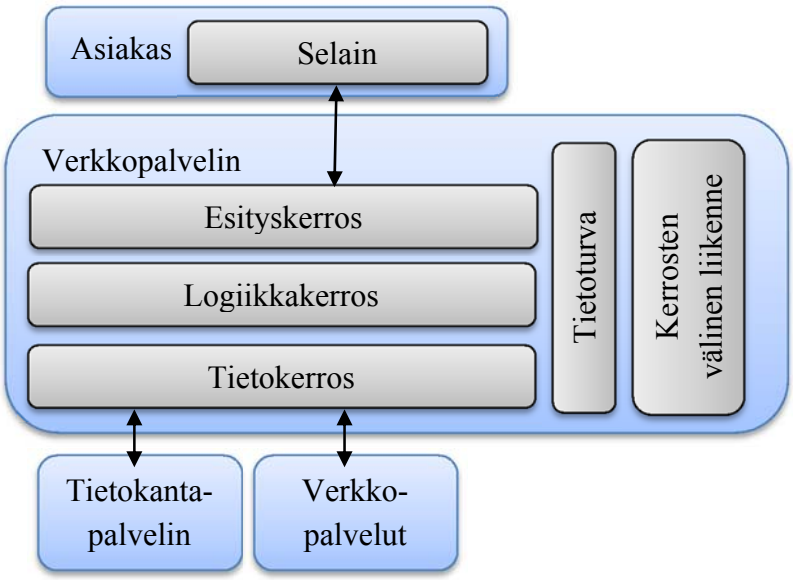
Asiakasohjelma lähettää HTTP-pyynnön verkkopalvelimelle, jonka otsakkeesta verkkopalvelin osaa määrittää mille verkkosovellukselle pyyntö ohjataan. Verkkosovellus saa pyynnön, käyttää resurssin luomiseen mahdollisesti tietokantoja tai muita resursseja, palauttaa vastauksen palvelimelle, joka lopulta palauttaa vastauksen asiakasohjelmalle. Koska verkkosovellukset ovat dynaamisia verkkosivuja, tarvitaan jokaisen palautettavan

HTML-tiedoston tuottamiseen laskentatehoa palvelimelta. Yhdellä verkkopalvelimella voi olla useita verkkosovelluksia, jotka jakavat keskenään palvelimen resursseja [8 s.5].

2.5.3 Verkkosovelluksen arkkitehtuuri ja tietokannat

Sovellussuunnittelua ja hajautettuja hypermediajärjestelmiä varten on kehitetty REST-arkkitehtuuri (REpresentational State Transfer). Jos WSDL ja SOAP ovat aikaisemmin olleet yleinen tapa suunnitella verkkopalveluja funktioilla ja operaatioilla, niin REST-suunnittelun tavoite on esitellä kaikki palvelut resursseina ja määritteillä tarkennetuilla aliresursseina. REST-suunnittelu toinen tavoite on suunnitella skaalautuvia, yhtenäisillä rajapinnoilla toimivia, itsenäisillä komponenteilla toteutettuja sovelluksia. Kun komponentit suunnitellaan itsenäisiksi, voidaan niiden tietoturvatestausta tehdä usealla tasolla, moduulien toimintaa nopeuttaa välimuistilla, sekä vanhoja komponentteja voidaan kapseloida vastaamaan päivittyviä vaatimuksia [44].

Arkkitehtuurista riippumatta, sovellukset koostuvat usein eri kerroksista, joista yleinen kolmikerroksinen esimerkki on kuvattu kuvassa 2.4.



Kuva 2.4 **Verkkosovelluksen rakenne kerroksittain**

Mallissa verkkosovellus on jaettu esitys-, logiikka-, ja tietokerrokseen. Esityskerros käsittelee tehtävät, jotka ovat vuorovaikutuksessa käyttäjän kanssa, kuten käyttäjäsyötteiden vastaanottamisen ja verkkosivujen esittämisen. Verkkosovellusten ydin on logiikkakerros, joka saa tietoja esityskerrokselta, suorittaa niille funktioita, pyytää mahdollisesti tietoja tietokerrokselta ja palauttaa muodostetun resurssin esityskerrokselle. Logiikkakerros suositellaan suunnittelemaan tilattomaksi, jolloin kaikki pyynnöt käsitellään aina samalla tavalla riippumatta aikaisemmista pyynnöistä tai käyttäjästä. Tietokerroksella käsitellään pitkäaikainen tieto, jota ei ohjelmoida valmiiksi logiikkakerroksen funktioihin.

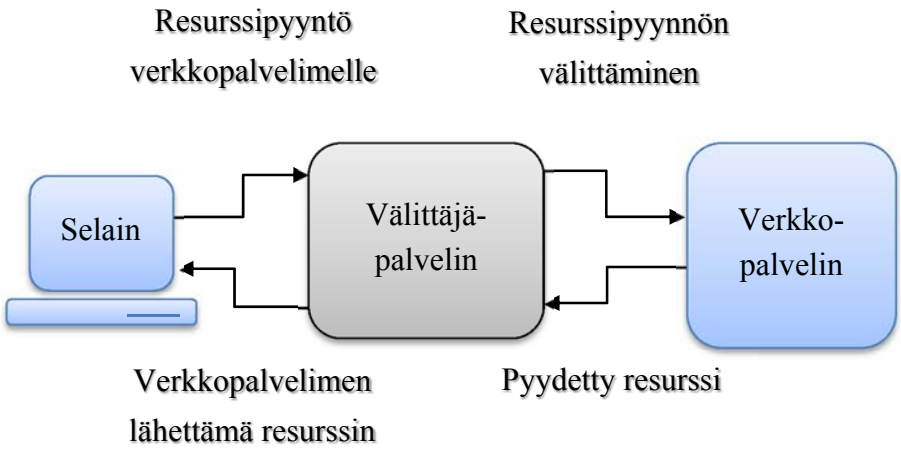
Tällöin tiedon päivittämiseen ja ylläpitoon voidaan käyttää tietokannan hallintajärjestelmää [8 s.14]. Sovelluksen jakaminen loogisiin kokonaisuuksiin helpottaa ylläpitoa, eri komponenttien toimivuutta voidaan tällöin tarkkailla itsenäisesti, sekä eri kerroksille voidaan asentaa välimuistia, jossa säilytetään yleisimmin tarvittavia tietoa. Edellä esitetyt tasot ovat yksi tapa suunnitella verkkosovellus ja eri toteutuksia on useita. Tietoturva testataan kerroskohtaisesti, sekä koko sovelluksen kattavana kokonaisuutena [17].

Verkkosovelluksista puhuttaessa käytetään usein termiä ”Backend”. Tällä viitataan verkkosovelluksen viimeiseen kerrokseen, jossa tietokantapalvelut yleensä sijaitsevat [8 s.16]. Tietokannat ovat loogisesti jäsenneltyjä tallennustiloja, joihin voidaan tallentaa tietoa ennalta määrätyssä muodossa. Kun tietokannassa olevien tietojen välille luodaan yhteyksiä, esimerkiksi avainkenttien avulla, puhutaan relaatiotietokannasta. Tietokannat voidaan yleisesti jakaa relaatiotietokantoihin ja NoSQL-tietokantoihin. Relaatiotietokantoja on käytetty tietotekniikassa jo 70-luvulla ja ne ovat yleisin tietokantamalli. Relaatiotietokantojen kanssa käytetään SQL-kyselykieltä (Structured Query Language) jonka komennot ovat ISO/IEC standardoitu. Termi NoSQL-tietokanta tarkoittaa kaikkia ei relaatiotietokantoja. Tällöin tietoalkiot eivät noudata ennalta määrättyjä taulukkomuotoja, vaan tietokannat järjestävät tietoa sen mukaan kun sitä tarvitaan. NoSQL-tietokannat ovat tehokkaampia suurien dynaamisten tietomäärien käsittelyssä, koska resursseja ei käytetä tietojen organisointiin ja relaatioiden ylläpitoon. Toisin kuin relaatiotietokannoilla, NoSQL-tietokannoilla ei ole yhteistä kyselykieltä tietokantakutsujen suorittamiseen, vaan komennot ovat tietokantasovelluskohtaisia. Jos tietokannan valinnan peruste on tietoturvallisuus, niin tällöin suositellaan relaatiotietokantaa [20].

Tietokantojen käyttöä tehostamaan on kehitetty tietokantojen hallintajärjestelmiä (engl. DataBase Management System, DBMS). Jos hallintajärjestelmä koskee relaatiotietokantaa, puhutaan relaatiotietokannan hallintajärjestelmästä (RDBMS). Näiden järjestelmien tarkoitus on optimoida tietokantoihin suoritettavia luku- ja kirjoitusoperaatioita, tarjota tukitoimia, kuten tietokannan varmuuskopiointia ja helpottaa tietokannan ylläpitoa esimerkiksi graafisen käyttöliittymän avulla. Yleisimpiä avoimen lähdekoodin relaatiotietokantojen hallintajärjestelmiä ovat SQLite, MySQL ja PostgreSQL [20]. Juuri tietokantojen käyttö on ollut tärkeä osa verkkosovellusten muuttumisessa staattisista sivuista dynaamisiksi sovelluksiksi.

2.5.4 Asiakas-palvelin -mallin häviäminen

Välittäjäpalvelin (engl. proxy) oli ensimmäinen askel pois asiakas-palvelin-mallista. Välittäjäpalvelin toimii verkossa tapahtuvien palvelinpyyntöjen välittäjänä, ja tämä on kuvattu kuvassa 2.5.



Kuva 2.5 Välittäjäpalvelimen toiminta

Asiakas lähettää pyynnön välittäjäpalvelimelle, samoin kuin lähettäisi sen verkkopalvelimelle. Välittäjäpalvelin lähettää pyynnön verkkopalvelimelle, joka palauttaa vastauksen välittäjäpalvelimen kautta käyttäjälle. Välittäjäpalvelimen avulla voidaan keventää verkkopalvelimen kuormaa, varastoida yleisimpiä palvelinpyyntöjä välimuistiin ja suodattaa verkkoliikennettä. Yhdellä verkkopalvelimella voi olla useita välityspalvelimia. Välittäjäpalvelin voi sijaita myös käyttäjän ja verkkoyhteyden välissä, jolloin välittäjäpalvelin voi toimia palomuurina ja estää määritettyjen verkkosivujen käytön. Välittäjäpalvelimien käytön yhteydessä alkuperäisen käyttäjän IP-osoite saattaa jäädä piiloon, koska pyyntö tulee välityspalvelimen IP-osoitteesta [8 s.16-17]. Hyökkääjän kannalta tämä on hyvä asia ja internetistä löytyy tällä tekniikalla toteutettuja palveluita, joiden tarkoitus on piilottaa alkuperäisen resurssipyynnön lähettäjä. Nykyaikaiset välityspalvelimet usein lisäävät HTTP-pyyntöön ylimääräisen otsakkeen, josta selviää alkuperäisen käyttäjän IP-osoite.

Käyttäjämäärien kasvaessa verkkosovellus voidaan jakaa useammalle identtisesti asennetulle verkkopalvelimelle. Kuormantasaaja toimii kuin käänteinen välittäjäpalvelin. Verkkosovellukselle tulevat pyynnöt kulkevat kuormantasaimen kautta, joka seuraa eri verkkopalvelimien kuormitusastetta ja ohjaa pyynnön eteenpäin palvelimilla olevan rasituksen, tai muun mittarin perusteella [8 s.16-17]. Toinen tapa keventää palvelimien kuormaa, on siirtää osa laskennasta selaimen suoritettavaksi. Ei ole resursoinnin kannalta järkevää, että verkkopalvelin on synkronisesti mukana jokaisessa selaimessa tapahtuvassa muutoksesta. Teknisesti verkkosovelluksen ei tarvitse olla kuin tietokantapalvelu verkossa ja selain voi suorittaa ohjelmakoodin jäsentämisen, suorittamisen ja esittämisen. Selaimeen voidaan lisäksi asentaa tietokantana toimivia laajennuksia, jolloin edes jatkuva yhteys verkkopalvelimeen ei ole pakollinen käytön aikana.

Tietoturva mielessä tehtävien jakaminen eri toimijoille aiheuttaa uusia ongelmia. Vaikka laskenta siirretään selaimelle, jää vastuu tietoturvasta sovelluksenkehittäjille. Koska laskentaa ei ole järkevää suorittaa vain palvelimelle, eikä sitä voida täysin siirtää selaimen suoritettavaksi, tekevät monet sovellukset kompromisseja tällä välillä. Ratkaisut ovat sovelluskohtaisia ja testaus vaikeaa, koska esimerkiksi palvelinta testattaessa täytyy

tietää miten selain käyttäytyy. Lisäksi tietokantasovelluksissa sijaitsee toisien käyttäjien tietoja ja tällöin sopivien oikeuksien antaminen on haasteellista. Yhtenäisten mallien ja testausprotokollien puuttuminen vaikeuttaa työtä ja tämä on yksi syy miksi verkkosovellukset sisältävät haavoittuvuuksia [5 s.17-18].

2.5.5 Selainpään ohjelmointi

Selainpään ohjelmointikielet mahdollistavat ohjelmakoodin suorittamisen selaimessa palvelimen sijasta. Ohjelmakoodi upotetaan osaksi HTML-dokumenttia tai se sijaitsee erillisessä tiedostosta, johon viitataan HTML-dokumentin otsikkokentässä. Selainpään ohjelmointia käytetään HTML-dokumentin muokkaamiseen, viestien esittämiseen, uusien välilehtien avaamiseen yms. Yleisin selainpään ohjelmointikieli on JavaScript. Ensimmäinen selaimeen upotettava Mocha-niminen ohjelmointikieli esiteltiin Netscape Navigatorille vuonna 1995. Toisessa julkaisuversiossa nimi vaihdettiin LiveScriptiksi ja yhteistyö Sun Microsoftin kanssa johti lopulta JavaScript-nimen valintaan. JavaScriptillä ei juurikaan ole tekemistä Java-ohjelmointikielen kanssa, mutta nimi valittiin markkinointi syistä. Microsoft esitteli oman VBScriptinsä 1996 osana IE 3.0-selainta, mutta sen myöhäinen markkinoille tulo ja kilpailijaa sekavampi syntaksi ei pystynyt horjuttamana JavaScriptin jo vakiintunutta asemaan. Toisin kuin useissa muissa selaintekniikoissa, JavaScript ei ole saanut myöhemmin merkittäviä kilpailijoita ja on ainoa merkittävä standardi selainpään ohjelmointikielissä [5 s.95-96].

JavaScript vaikuttaa verkkosivun ulkoasuun dokumenttioliomallin kautta. Dokumenttioliomalli on ohjelmointirajapinta, jonka avulla voidaan kutsua HTML-dokumenttien elementtejä, sekä lukea ja asettaa niiden arvoja ja parametreja [5 s.109-110]. Jokainen selaimessa esitettävä HTML-dokumentti on oma erillinen nimiavaruutensa, jonka sisällä JavaScript toimii. Dokumentit voivat vaihtaa tietoa keskenään, mutta tämä täytyy tehdä ennalta määrätyillä metodeilla. Jokaisen dokumentin sisällä oleva JavaScript-lohko suoritetaan itsenäisesti esiintymisjärjestyksessä. Yksinkertainen JavaScript-esimerkki on kuvattu ohjelmassa 2.5.

```
<script>
function display_string(str_teksti) {
alert(str_teksti);
return 0;
}

// Kutsutaanfunktiota joka esittää tekstin “Hello World!”.
var teksti = "Hello World!";
display_string(teksti);
</script>
```

Ohjelma 2.5 **JavaScript esimerkki**

Esimerkin alussa esitellään funktio, joka esittää kutsuttaessa sille parametrina annetun merkkijonon ja esimerkin lopussa kutsutaan tätä funktioita. JavaScript-lohkot rajataan <script>-tunnisteilla. Kun HTML-dokumentissa renderöintimoottori saavuttaa JavaScript-lohkon, aloitetaan ohjelmakoodin jäsennysvaihe. Tämä vaihe tarkistaa syntaksin oikeamuotoisuuden ja kääntää ohjelmakoodin nopeammin suoritettavaan binäärimuotoon. Jäsennys suoritetaan JavaScript-lohko kerrallaan, eikä lohkoja ketjuteta yhdeksi ohjelmakoodiksi. Jos jokin lohko, ei ole syntaksin mukainen, lopetetaan sen jäsentäminen ja siirrytään HTML-dokumentissa eteenpäin. Jos JavaScriptin ohjelmakoodi on syntaksin mukaista, aloittaa selainohjelma sen suorittamisen. Suorituksen aikana selain, tai ainakin sen renderöintimoottori, ei pysty muihin operaatioihin. Suorituksen aikana tulleet pyynnöt suoritetaan vasta käynnissä olevan ohjelmakoodin suorittamisen jälkeen. Jos ohjelmakoodissa on huonosti suunniteltua, esimerkiksi sisältää loputtomia silmukoita, täytyy selaimen keskeyttää sen suorittaminen ja tähän on erilaisia tekniikoita selainkohtaisesti [5 s.97-101].

2.5.6 Julkaisujärjestelmät

Muutos staattisista sivuista dynaamisiksi verkkopalveluiksi on johtanut julkaisujärjestelmien, viitekehysten ja kotisivukoneiden suosion tasaiseen kasvuun. Vuonna 2013 San Franciscossa Wordpress-julkaisujärjestelmän luoja Matt Mullenweg kertoi, että 18,9 prosenttia kaikista verkkosivuista käyttää alustanaan Wordpress-julkaisujärjestelmää [21]. Samansuuntaiseen tulokseen on tullut W3Techs, jonka tilasto vuodelta 2014 on esitetty taulukosta 2.1. Taulukossa on esitetty viiden suosituimman julkaisujärjestelmän osuudet kaikista verkkosivuista ja julkaisujärjestelmien keskinäisistä markkinaosuuksista.

Taulukko 2.1 - Julkaisujärjestelmien suosio vuonna 2014

Ohjelmisto	Osuus kaikista sivuista	Markkina osuus
WordPress	21.9%	60.3%
Joomla	3.1%	8.6%
Drupal	1.9%	5.3%
Blogger	1.1%	3.1%
Magento	1.0%	2.6%

Taulukosta huomataan, että WordPress on selvästi suosituin julkaisujärjestelmä. Syyksi on arvioitu matalaa oppimiskynnystä sekä kuluttajien ja pienyritysten suosiota, kun taas Joomla! ja Drupal ovat laajemmin räätälöitävissä olevia yrityksille suunnattuja sovelluksia. Kun lasketaan kaikkien julkaisujärjestelmien osuudet yhteen, huomataan että yli 30 prosenttia kaikista verkkosivuista on toteutettu käyttäen julkaisujärjestelmiä [13]. Syitä suosioon ovat valmiit ratkaisut sivujen ylläpitoon, käyttäjien hallintaan, helposti asennettavat lisäosat, sekä järjestelmien laaja osaaminen ja tuki. Räätälöityjen palveluiden ja itse ohjelmoitujen verkkosovellusten osaaminen on usein keskittynyt yksittäisille henkilöille yrityksessä ja tällainen voidaan nähdä tietoturvariskinä [22].

Jeremiah Shoaf kirjoittaa blogissaan kiinnostavia argumentteja julkaisujärjestelmien tulevaisuudesta [23]. Hänen mukaansa raskaat tietokantapohjaiset julkaisujärjestelmät väistyvät tulevaisuudessa uusien matalien julkaisujärjestelmien tieltä. Matalat julkaisujärjestelmät eivät käytä tietokantoja, vaan tiedot tallennetaan verkkopalvelimelle sijaitseviin tiedostoihin ja kansioihin. Ratkaisu on nopeampi, yksinkertaisempi ja turvallisempi. Tiedostotasolla versiohallinta on helppo toteuttaa ja verkkosivut voidaan siirtää pelkästään kopioimalla tiedostot toiselle palvelimelle. Tietokantoja tarvitsevat osuudet, kuten kommenttikentät, luodaan kolmannen osapuolen verkkopalveluilla. Tällöin tietoturvaosaaminen ja palveluiden saavutettavuus pystytään ulkoistamaan. Hyvä esimerkki on verkkosivuille upotetut videot. Aikaisemmin video ladattiin verkkopalvelimelle ja ohjelmoijan tehtävä oli hallita videon koodauksen purkaminen eri päätelaitteille, selaimille, resoluutioille ja latausnopeuksille. Nyt sama palvelu voidaan tuottaa upotettuna elementtinä Youtube:n tai Vimeo:n kaltaisista palveluista.

2.6 Verkkohotellipalvelu

Tässä kappaleessa käsitellään mitä teknisiä komponentteja ja ominaisuuksia verkkohotellipalvelu sisältää. Palvelu on suunnattu asiakkaille, jotka haluavat ylläpitää verkkosovelluksia tai -sivuja, mutta eivät verkkopalvelinta. Syy voi olla kustannustehokkuus, osaaminen tai vastuun siirtäminen ylläpidosta palveluntarjoajalle [24]. Palvelua käytetään lisäksi sähköpostiliikenteen hoitamiseen. Asiakas on vastuussa palveluun asentamiensa ohjelmistojen ja lataamiensa tietojen tietoturvasta ja asiakkaan ja palveluntarjoajan vastuita käsitellään tarkemmin kappaleessa 4.

dHosting-nimen alla tarjotaan tällä hetkellä vain jaettua palvelintilaa. Asiakkaalle dedikoitu palvelut toteutetaan Decens Oy:n muusta tuotevalikoimasta ja ne on rajattu tämän diplomityön ulkopuolelle. Verkkohotellipalvelu on toteutettu fyysisillä palvelimilla, joiden päälle on asennettu virtualisointikerros (engl. hypervisor). Palvelimet sijaitsevat konesalista, jossa niille taataan sähkönsaanti, jäähdytys, tarvittava määrä verkkoresursseja, sekä fyysinen tietoturva. Resurssit ovat jaettu kahdelle virtuaalipalvelimelle, joista toisessa on Windows ja toisessa Linux-käyttöjärjestelmä. Molemmille palvelinalustoille on asennettu käyttöjärjestelmäkohtainen versio Parallels Plesk Panel-verkkohotellien hallintaohjelmistosta. Parallels Pleskin Panelin asennus sisältää verkkopalvelimen, tietokantaohjelman, sekä käyttöön tarvittavien komentosarjakielien asennukset. Verkkopalvelimet eivät operoi verkkosivujen kanssa, vaan ohjaavat käyttäjien pyynnöt virtuaalisille verkkohotelleille (engl. virtual host, vhost), jotka ovat teknisesti pieniä virtuaalipalvelimia. Verkkohotellien ominaisuudet määritellään palveluntarjoajan luomilla palvelusuunnitelmilla. Palvelusuunnitelmia voi olla useita ja ne määrittävät kuinka paljon asiakkaalle tarjotaan kotisivutilaa, verkkokaistaa, sähköpostiosoitteita yms. Verkkohotellit erotellaan toisistaan joko IP-osoitteen tai verkko-osoitteen avulla [25].

2.6.1 Verkkohotellin toteuttamisen vaihtoehdot

Yleisin tapa toteuttaa verkkohotellipalvelu on jaettu palvelintilan tarjoaminen (engl. shared hosting). Tällöin yhdellä verkkopalvelimella, ylläpidetään useita verkkosivuja. Kaikki verkkosivut jakavat yhteiset palvelinresurssit ja yksittäisen verkkosivun resurssien käyttö voi vaikuttaa muiden verkkosivujen toimintaan. Asiakkaat eivät saa palvelimelle pääkäyttäjätason oikeuksia, vaan rajatun tiedostonäkymän omaan kotisivutilaansa. Palvelun tarjoaminen voi olla IP-perusteista, jolloin jokaisella asiakkaalla on oma IP-osoite. Yleisempi ratkaisu on nimi-perusteinen palvelintilan jakaminen. Tällöin palvelun asiakkaat jakavat saman IP-osoitteen, ja erottelu tehdään HTTP-pyynnön otsakkeessa olevan verkko-osoitteen. Palvelu on teknisesti yksinkertaisempi ja se säästää tarvittavia IP-osoitteita. Kaikille palvelussa oleville asiakkaiden verkko-osoitteille märitellään samat nimipalvelintiedot ja tämän jälkeen verkkopalvelin ohjaa pyynnöt oikeille verkkohotelleille verkkopalvelimen zone-asetusten perusteella [26]. Ohjelmassa 2.6 on esitetty, kuinka Apache-verkkopalvelimella tehdään nimiperusteiseen jakamiseen tarvittavat zone-asetukset httpd.conf-tiedostoon, käyttäen NameVirtualHost-asetusta.

```
NameVirtualHost *:80

<VirtualHost *:80>
ServerName www.osoite1.fi
DocumentRoot /www/osoite1
</VirtualHost>

<VirtualHost *:80>
ServerName www.osoite2.fi
DocumentRoot /www/osoite2
</VirtualHost>
```

Ohjelma 2.6 **Apache-verkkopalvelimen zone-asetusten asettaminen kahdelle eri verkko-osoitteelle**

Esimerkissä on kaksi verkko-osoitetta: osoite1.fi ja osoite2.fi. Jokaiselle verkkohotellin asiakkaalle luodaan oma <Virtualhost>-argumentti, joka yhdistää palvelimen IP-osoitteen porttiin 80 tulleet HTTP-pyynnöt otsakkeen perusteella oikeaan tiedostokansioon. Tiedostokansion juuressa on usein index.html-niminen tiedosto, joka esitetään verkko-osoitteen aloitussivuna [27].

Dedikoitu palvelin (engl. dedicaded hosting, dedicaded server) on palvelu, joka toteutetaan fyysisellä palvelimella ja jonka resurssit ovat vain yhden asiakkaan käytössä. Tämä on usein kallein ratkaisu verkkohotellipalvelun toteuttamiselle. Tällöin asiakas voi vaikuttaa palvelimen fyysisten komponenttien valintaa ja hänelle voidaan myöntää pääkäyttäjäoikeudet käyttöjärjestelmään. Asiakas voi itse omistaa palvelimen tai se voi olla vuokralla palveluntarjoajalta. Fyysisten palvelimen vuokraaminen on vähentynyt ja nimetyt palvelinratkaisut toteutetaan nykyään virtuaalisina instansseina. Tällöin käytetään

ermiä virtualisoitu yksityinen palvelin (engl. virtual private server) [26][15 s.26-27]. Virtuaalisissa ratkaisuissa asiakkaan palvelinkapasiteetti voidaan muokata joustavammin ja palvelu on edullisempi, kuin fyysisen palvelimen omistaminen. Palveluntarjoajan edut virtualisoinnissa ovat keskitetty ylläpito ja palvelinresurssien hukkakapasiteetin väheneminen.

Verkkohotellipalvelun ja palvelutilan tarjoaminen toteutetaan usein pilvitoteutuksena, joka tarkoittaa useiden palvelimien liittämistä näennäisesti yhdeksi palvelinklusteriksi. Palvelimien ei tarvitse sijaita samassa konesalissa, vaan ne yhdistyvät verkon yli loogiseksi kokonaisuudeksi. Kun palvelimet ovat hajautettu maantieteellisesti, saadaan palvelut lähemmäksi käyttäjiä sekä paikkasidonnaiset riskit vähenevät. Yhden fyysisen palvelimen rikkoutuminen laskentaklusterista hidastaa palvelua, mutta palvelu saatavuus ei esty. Pilvitoteutuksen huonot puolet liittyvät tietoturvariskeihin, kuten tiedon paikkasidonnaisuuden menettämiseen ja siihen, että tietomurto yhteen palvelimeen voi vaarantaa koko pilven tietoturvan [15 s.26].

2.6.2 Verkkohotellin hallintapaneeli

Verkkohotellin hallintapaneeli (engl. hosting control panel) tarkoittaa palveluntarjoajan verkkopalvelimella asentamaa verkkohotellien hallintaohjelmaa. Yleisimpiä hallintapaneeleja markkinoilla ovat cPanel ja Plesk Panel. Pienet verkkohotellipalveluntarjoajat usein ostavat hallintapaneeliohjelmistonsa, mutta suuret palveluntarjoajat saattavat toteuttaa ohjelmiston itse. Oman toteutuksen etuja ovat säästetyt lisenssikulut ja mahdollisuus laajempaan hallittavuuteen ja kehitykseen. Ostettujen ohjelmien etuja ovat matala käyttöönottokynnys ja valmistajan tuki ongelmatilanteissa [15 s.30-31].

Plesk-paneeli toimii komentojen välittäjänä käyttäjän ja verkkopalvelimien välillä. Kun ylläpitäjä luo uuden asiakkaan hallintapaneelissa, tarvittavat komennot välittyvät automaattisesti verkkopalvelimelle, joka luo uuden verkkohotellin. Hallintapaneelin kautta hallitaan palvelimen verkko-, DNS-, ja sähköpostiasetuksia, sekä luodaan sähköpostilaatikoita, FTP-tunnuksia, tietokantoja ja aliverkko-osoitteita. Oikeudet näihin tehtäviin voidaan rajoittaa palvelun ylläpidolle tai myöntää lisäksi palvelun asiakkaille. Hallintapaneeli tarjoaa lisäksi tilastotietoa verkkosivujen toiminnasta, resurssien käytöstä, sekä antaa palveluntarjoajalle tietoa myynnin, laskutuksen ja kehityksen tehtäviin. Hallintapaneelia käytettäessä on tärkeä ymmärtää, että teknisesti se on verkossa toimiva graafinen verkkopalvelimen hallintatyökalu ja siinä tehtävät muutokset tapahtuvat verkkopalvelimen asetuksissa, ei paneelin omissa asetuksissa [29]. Parallels Plesk Panel sisältää lisäksi XML-over-HTTP-pohjaisen RPC-rajapinnan(Remote Procedure Call), joka mahdollistaan paneelin komentopohjaisen hallinnan. Rajapinnasta on oma versio sekä Linux että Windows-käyttöjärjestelmille ja sen kautta voidaan automatisoida esimerkiksi asiakkaiden luonti tai liittää paneelissa suoritettavat toiminnot osaksi yrityksen toiminnanohjausjärjestelmää. Rajapintaan käytetään samoilla käyttäjätunnuksilla kuin graafista hallintapaneelia, joten myös asiakas pystyy automatisoimaan omia toimintojaan verkkohotellissa [28].

2.6.3 Sähköpostipalvelu osana verkkohotellia

Domain-päätteiset sähköpostiosoitteet ovat osa yrityksen imagoa ja kiinteä osa verkkohotellipalvelua. Sähköpostitilien hallinta tapahtuu verkkohotellin hallintapaneelin kautta ja sähköpostiosoitteiden määrä on riippuvainen ostetusta palvelusuunnitelmasta. Sähköpostipalvelu käyttää samaa kotisivutilaa verkkohotellipalvelun kanssa ja sähköpostilaatikon koko voidaan määrittää käyttäjä- ja ryhmäkohtaisesti. Sähköpostien hallintaan käytetään sähköpostiohjelmaa tai verkossa toimivaa asiakasohjelmaa. Plesk-hallintapaneelista voidaan luoda sähköpostilistoja, sekä hallita sähköpostien ohjauksia [15 s.39-54].

Sähköpostipalvelu ja verkkosivut voidaan hajauttaa kahdelle eri palvelimelle tai sähköpostipalvelu voidaan ulkoistaa eri palveluntarjoajalle. MX-taulukko (engl. mail exchange record) on ohjaustaulukko, joka määrittää minkä palvelimen kautta sähköpostiliikenne reititetään. Etuna HTTP-liikenteen ja sähköpostiliikenteen erottamisessa eri palvelimille on riskien hajauttaminen. Sähköpostipalvelut, kuten Googlen Gmail, mahdollistavat sähköpostiliikenteen ohjauksen Googlen sähköpostipalvelimien kautta. Asiakas hallitsee verkkohotellipalvelun MX-taulukon asetuksia Parallels Plesk hallintapaneelin kautta [15 s.58-60].

2.6.4 PHP ja muut komentosarjakielet

PHP (Hypertext Preprocessor) on verkkosovellusten ohjelmointiin käytetty komentosarjakieli ja sen avulla voidaan välittää tietoa selaimelta palvelimella suoritettavalle sovellukselle. Parallels Plesk-hallintapaneelin ylläpitäjän opaskirjan mukaan suurin osa dynaamisista verkkosivuista ja verkkopalveluista perustuu PHP-kielisiin ohjelmakoodeihin. W3Techsin vuonna 2014 tekemän tutkimuksen mukaan jopa 81,9 prosenttia palvelinpäänohjelmointia suorittavista verkkosivuista, käyttävät PHP-ohjelmointikieltä [30]. Seuraavina listalla pienemmillä osuuksilla ovat ASP.NET, Java, ColdFusion, Perl, Ruby ja Python [13]. PHP-tulkki voidaan asentaa IIS ja Apache verkkopalvelimille ja se on yleisesti valmiiksi asennettu verkkohotellipalveluntarjoajien verkkopalvelimille [31].

PHP-ohjelmien käyttäytyminen palvelimella riippuu kolmesta asiasta, joista ensimmäinen on PHP-tulkki. PHP-kieliset pyynnöt käsitellään palvelimella sivun pyynnön yhteydessä ja tätä käännöstä varten tarvitaan PHP-tulkki [31]. dHosting palveluun on asennettu useita eri tulkkeja, kuten ISAPI, FastCGI, CGI tai PHP-FPM. Mitä tulkkia suositellaan käytettäväksi, riippuu ohjelmakoodin kääntämisnopeudesta, muistin käytöstä ja turvallisuusvaatimuksista. Toinen merkitsevä asia on PHP-versio. Useimmat PHP-versiot eivät ole taaksepäin yhteensopivia ja Plesk paneeli tukee useamman PHP-version yhtäaikaista asennusta, joista käytettävä versio voidaan määritellä verkkohotellikohtaisesti. W3Techs-järjestön tutkimuksen mukaan PHP-versio 5 on käytössä 97,8 prosentilla verkkosivuista, joten eri versioiden ja taaksepäin yhteensopivuuden merkitys on kohtuullisen pieni. Kolmas vaikuttava asia on PHP-asetukset. Asetuksista määritellään esimerkiksi ohjelmakoodin muistirajoitukset, minkä kokoisia tiedostoja palvelimelle voidaan ladata, monta resurssipyyntöä voidaan suorittaa tietyn aika ikkunan sisällä, yms. Näillä rajoituksilla estetään huonosti kirjoitettujen ohjelmien muistivuodot ja suojaudutaan vihamielisiä

ohjelmia vastaan. Huonosti kirjoitettu ohjelmakoodi ja huonot PHP-asetukset voivat häiritä koko verkkohotellipalvelun toimintaa [30][32].

2.6.5 Asiakkaat

HostGator on yksi maailman suurimmista verkkohotellipalveluiden tuottajista ja sen perustaja Abrahim Ikasud kirjoittaa uuden palvelun perustajan ohjekirjassa, että kaikki asiakastilaukset kannattaa käsitellä manuaalisesti [33]. Palvelun tuottamiseen käytettävä ohjelmisto sallii palveluiden käyttöönoton automatisoinnin, mutta tämä avaa kanavan mahdolliselle väärinkäytölle. Abrahamin oman kokemuksen mukaan juuri vihamielisillä käyttäjillä on suurin kiire, koska heidän on esimerkiksi juuri suljettu pois toiselta palveluntarjoajalta. Pitkäaikaista ja kestävää ratkaisua hankkiva asiakas odottaa palvelun käyttöönottoon kuluvan ajan. Samalla palveluntuottaja pysyy paremmin tietoisena asiakaskunnastaan.

Kun tilaukset käsitellään käsin, voidaan tilaukset analysoida lisäksi sisällön perusteella. Jos tilauksen verkko-osoite muistuttaa esimerkiksi tunnetun pankin nimeä tai siinä on viitteitä aikuisviihteeseen, voidaan se tutkia tarkemmin. Lisäksi verkko-osoitteiden rekisteröintiä ei pidä automatisoida, koska se on palveluntarjoajalla maksullinen toimenpide. Verkkohotellin avaaminen hallintapaneeliin maksaa palveluntarjoajalle vain työntekijän käytetyn ajan. Muita mahdollisia tilauksille suoritettavia tarkistuksia ovat pankkikorttimaksujen laillisuus, onko tilaajan puhelinnumero ja osoite olemassa, tai onko tilauksen tehneen yrityksen tiedot kunnossa. Samoin välittäjäpalvelimien kautta tulleet tilaukset on hyvä aina tarkistaa tai kieltää kokonaan.

3 TIETOTURVA JA TIEDON OMINAISUUDET

Tässä luvussa käsitellään mitä termi tietoturva tarkoittaa ja miten se liittyy yritysten tietoturvapolitiikkaan. Tietoturvalle on todellista tarvetta ja yritysten on otettava se huomioon toiminnassaan. Englannin kautta- ja teollisuusministerin vuonna 2006 tekemän tutkimuksen mukaan maan yrityksistä 62 prosenttia on kokenut ongelmia tietoturvan kanssa. Keskiarvo tietoturva ongelmille oli kahdeksan kappaletta vuodessa. Yleisimpiä ongelmia olivat virukset (35 prosenttia), henkilökunnan tiedon väärinkäyttö (21 prosenttia) ja ulkoa tulleet hyökkäykset (17 prosenttia). Yritykset jotka joutuivat verkosta tulleiden hyökkäyksien kohteeksi, saattoivat saada satoja hyökkäyksiä päivässä. Hyökkäysten määrä on luultavasti suurempi, koska yritykset pitävät osan tietoturvaongelmista salassa yrityksen imagon säilyttämiseksi [9 s38]. Tulos on linjassa Matti Laakson arvion kanssa, jonka mukaan tietoturvasta 20 prosenttia on teknistä suojautumista ja loput 80 prosenttia työntekijöiden kouluttamista ja ongelmia yrityksen sisäisissä prosesseissa [34].

Tiedon arvon määrittäminen on usein vaikeaa ja sen ylläpitoon on täytynyt kehittää uusia menetelmiä. Kirjassa Security in computing (Pfleeger 2006) käsitellään tilannetta pankin näkökannalta. Ennen varat olivat talletettu pankkeihin ja ryöstäjän täytyi mennä paikan päälle, ohittaa fyysiset suojaukset ja kuljettaa varastamansa tuotteet pois paikalta. Tietovarkauksissa hyökkääjän fyysisellä sijainnilla ei ole merkitystä, hyökkäystä ei mahdollisesti edes huomata ja suurien tietomäärien varastaminen tapahtuu hetkessä. Lisäksi sähköisen tuotteen kopioiminen ja myyminen useammalle asiakkaalle on mahdollista. Rikos ja sen kohde usein sijaitsevat maantieteellisesti eri alueilla ja tällöin sovellettavan lainsäädännön tulkitseminen on vaikeaa. Uutisoidut tietomurrot vaikuttavat negatiivisesti kohteeksi joutuneen yrityksen imagoon, mutta positiivisena puolena voidaan nähdä, että tietomurron kohteeksi joutuneella yrityksellä voi olla vahva tahtotila hoitaa tietoturvaan liittyvät asiat kuntoon [35 s.5].

Suurin osa verkkohyökkäyksistä tulee automatisoiduilta ohjelmilta (engl. bots), jotka etsivät sovelluksista ennalta tunnettuja tietoturva-aukkoja. Vakavampi uhka ovat kohdennetut hiljaiset hyökkäykset. Kohdennetuissa hyökkäyksissä tietoa kohdejärjestelmästä voidaan kerätä ja ostaa jopa vuosia ajan ja teho perustuu räätälöintiin kohteen mukaan. Virusohjelmat ja useat vastatoimet perustuvat usein tunnettujen hyökkäyksien havaitsemiseen ja eivät tunnista räätälöityjä hyökkäyksiä. Amerikkalaisen tutkimuksen mukaan 94 prosenttia onnistuneista tietoturvahyökkäyksistä tulee yrityksen tietoon yrityksen ulkopuoleisista läheistä. Tämä tarkoittaa, että yritysten omat järjestelmät havaitsevat vain 6 prosentti hyökkäyksistä, jotka onnistuvat tunkeutumaan tietojärjestelmiin. Huomaamatta jääneiden hyökkäysten määrää on todennäköisesti suuri, koska useat onnistuneet hyökkäykset ovat kestäneet jopa vuosia ennen niiden havaitsemista. Suomen vastaavia lukuja voidaan vain arvailla [36][37].

Tietoturvavaatimukset vaihtelevat yrityksissä projektikohtaisesti. Henkilöstölle tehtävät taustatarkistukset ja vaitiolosopimukset ovat normaaleja toimenpiteitä. Hankkeiden selvitysvaiheessa voidaan arvioida ympäristön vaikutukset, työtilojen turvallisuus, tehdäänkö palvelu omissa vai asiakkaan tiloissa, sekä maantieteellinen sijainti ja sen vaikutukset. Henkilöiden nimeäminen tietoturva-tehtäviin selkeyttää projekteja ja laskee työntekijöiden kynnystä kysyä ja ottaa selvää asioista. Uusien palveluiden täytyy vastata yrityksen nykyistä tietoturvapolitiikkaa, mutta vanha tietoturvapolitiikka ei saa olla esteenä kehitykselle ja sitä täytyy voida muokata tarpeen ilmetessä. Tällöin kokonaiskuva tietoturvasta paranee ja ratkaisut yhtenäistyvät. Huonot ja sekavat ohjeet voivat häiritä yrityksen toimintaa enemmän, kuin ei ohjeita ollenkaan. Jos eri palveluilla ja projekteille on eri käytännöt, sekaannuksien ja virheiden todennäköisyys kasvaa [38].

3.1 Tietoturvan peruskäsitteet

Tässä kappaleessa käsitellään tietoturvan peruskäsitteitä, kuten tiedon ominaisuuksia, uhkia ja haavoittuvuuksia sekä mahdollisia kohteita. Tarkoitus on omaksua teorian perusteita ja termejä, jotta niitä voidaan käyttää hyväksi myöhemmin suoritettavassa tietoturvatestauksessa.

3.1.1 Tiedon ominaisuudet

Tietoturvasta puhuttaessa tieto jaotellaan ominaisuuksiin luottamuksellisuus, eheys ja saatavuus. Samat ominaisuudet pätevät sekä palveluihin että konkreettiseen tietoon, kuten dokumentteihin tai tietokantoihin.

Luottamuksellisuus tarkoittaa, että tietoa voi käsitellä vain sellaiset henkilöt, joilla on siihen oikeus. Luottamuksellisuus ei tarkoita ainoastaan tiedoston lukemista, vaan myös kirjoitus-, muutos-, tulostusoikeutta ja sitä, että käyttäjä on edes tietoinen hyödykkeen olemassaolosta. Yksityisyyttä ja tiedon salausta kutsutaan myös luottamuksellisuudeksi. Luottamuksellisuutta määritellään käyttäjäoikeuksilla, kuten saako käyttäjä oikeudet vain ennalta määrättyihin tietoihin, koko tietokantaan ja saako käyttäjä kopioida tietoja tai jakaa niitä eteenpäin. Luottamuksellisuus on helposti ymmärrettävä määre, koska siitä löytyy normaaliin elämään helposti samaistettavia esimerkkejä [38][35 s.29-31].

Eheys tarkoittaa, että tieto ei saa muuttua tahattomasti tai sen täytyy muuttua tietyn prosessin mukaisesti. Muuttumisella tarkoitetaan tiedon muokkaamista, luomista tai poistamista. Jos tieto muuttuu, niin tämä täytyy pystyä havaitsemaan. Eheyden hallintaan kuuluvat prosessit, joilla tieto saa muuttaa ja ne henkilöt, joilla on tähän oikeus [38][35 s.29-31].

Saavutettavuus, käytetään myös termiä saatavuus, tarkoittaa, että tieto tai tiedon saantiin tarvittava palvelu on käytettävissä ennalta määrättynä ajanjaksona. Saavutettavuus tunnetaan usein sen vastakohdasta, eli palvelun estosta. Palvelu voidaan sanoa olevan saavutettava, kun sen sisältämä tieto on esitetty selkokielisessä muodossa, palvelun kapasiteetti täyttää tilaajan tarpeen ja palvelu voidaan toimittaa ennalta määrätyssä ai-

kaikkunassa. Lisäksi palvelun resurssit on jaettu ennalta määriteltyjen perusteiden mukaisesti kaikille palvelua pyytäville tahoille, sekä tiedon rinnakkainen käyttö on mahdollista. Virhetilanteet eivät saa aiheuttaa tiedon menettämistä ja tiedon saantiin on ongelmatilanteissa varasuunnitelma [38][35 s.29-31].

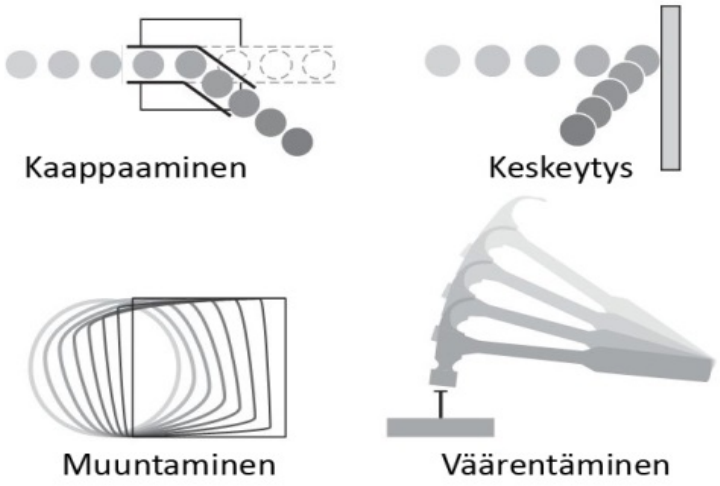
Tietoturvan tarkoitus on turvata näitä ominaisuuksia. Ominaisuuksia voidaan tarkastella lisäksi niiden hyökkääjän näkökulmasta. Luottamuksellisuus takaa, että tiedon olemassaolo ei paljastu ulkopuolisille, eheys takaa, että tietoa ei muokata luvattomasta ja saatavuus takaa, että tiedon saanti ei esty. Haaste järjestelmää tai palvelua suunniteltaessa on löytää tasapaino ominaisuuksien kesken. Kaksi kolmesta ominaisuudesta on helppo taata. Täydellinen eheys ja luottamuksellisuus voidaan toteuttaa järjestelmällä, johon kukaan ei pääse käsiksi ja siinä olevia tietoja ei voi muuttaa. Järjestelmä ei kuitenkaan ole tietoturvallinen, koska saavutettavuus ei toteudu. Tietoturvallisuus keskittyy usein näihin kahteen osa-alueeseen saatavuuden ja käytettävyyden kustannuksella [35 s.30-32;37].

Palvelun tietoturvan kuvauksessa käytetään myös lisämääreitä. Jäljitettävyyden avulla järjestelmästä voidaan selvittää mitä järjestelmässä on tehty, kuka on mahdollisen muutoksen tekijä ja milloin muutos on tehty. Muita täydentäviä ominaisuuksia on kiistämättömyys, tunnistus sekä todennus [38]. Jotta tietoja voidaan käsitellä oikein, täytyy se lajitella tietoturvatason mukaisesti. Yrityksen päätettäväksi jää, miten tiedot luokitellaan. Tästä karkein jako on julkiset tiedot ja salaiset tiedot. Tämä jako on usein liian suppea, ja lisäksi käytetään luokkia sisäiset tiedot ja luottamukselliset tiedot [39 s.26].

3.1.2 Uhkat ja haavoittovuudet

Code Complete-kirjassa Steve McConnell on tutkinut virheiden määrää tuotettua koodiriviä kohden. Kaupallisessa koodissa virheitä on 15-50 kappaletta jokaista tuhatta riviä kohti. Microsoftin ohjelmissa on testausvaiheessa 10-20 virhettä ja lopullisessa tuotteessa 0,5 virhettä jokaista tuhatta riviä kohti. Ohjelmistotestauksen erikoistyökaluilla päästään tulokseen 0,1 virhettä jokaista tuhatta riviä kohti. Päätelmä on, että ohjelmissa ja järjestelmissä on virheitä ja virheet voivat olla haavoittuvuuksia. Jokainen haavoittuvuus on uhka ja tietoturvan yksi tehtävä on estää näiden uhkien toteutumasta. Yksi tapa suojautua hyökkäyksiltä on estää uhkien tulemista julki [41][35 s.29].

Tietoturva voidaan määritellä käsitteillä haavoittuvuus, uhka, hyökkäys ja vastatoimet. Haavoittuvuus on järjestelmässä oleva väärinkäytön mahdollistava heikko kohta, kuten koodivirhe, avoin verkkoportti, heikko salasana tai kouluttamaton käyttäjä. Uhka on haavoittuvuutta hyväksi käyttävä tapahtuma, joka aiheuttaa ei toivotun tilan järjestelmään. Uhkia ovat esimerkiksi huolimaton käyttäjä, joka poistaa tietoja järjestelmästä tai varmistamaton palvelin joka hajoaa. Hyökkäys on tilanne, jossa hyökkääjä käyttää järjestelmän haavoittuvuutta hyväkseen. Vastatoimi tarkoittaa varautumista epätoivottuihin tilanteisiin. Voidaan sanoa, että uhkat vältetään haavoittuvuuksien vastatoimilla. Uhkien luokitteluun käytetään eri kategorioita. Yksi tapa luokitella uhkat on Pfleegerin luokitus vuodelta 2009, joka esitetään kuvassa 3.1. Jaottelu on vanha, mutta sitä käytetään vielä yleisesti.



Kuva 3.1 **Eri uhkaluokitukset jaoteltuina eri kategorioihin [35 s.8]**

- Kaappaaminen tarkoittaa tiedon tai resurssin luvatonta käyttöä tai haltuun ottoa
- Keskeytys tarkoittaa oikeutetun toiminnon keskeyttämistä tai palvelun saattamista pois toimintakunnosta
- Muuntaminen tarkoittaa alkuperäisen oikean tiedon muuntamista toiseen muotoon
- Väärentäminen tarkoittaa kokonaan uuden resurssin tai tiedon syöttämistä järjestelmään [35 s.6-8;10;15]

Vastatoimet tarkoittavat haavoittuvuuksien peittämisen lisäksi hyökkäyksien estämistä, sekä niiden seurauksien, eli riskien hallintaa. Riski voidaan määritellä seuraavalla kaavalla:

$$Riski = Uhkan\ todennäköisyys * Uhkan\ seuraukset$$

Kaavasta huomataan, että hyökkäyksen todennäköisyyden pienentäminen on vain osa tietoturvasta. Myös niiden seuraukset täytyy tuntea, että oikeat riskin voidaan määritellä ja niihin osataan varautua. Uhkan seurauksien määrittelyyn on useita työkaluja ja niillä arvioidaan riskien sekä teknillistä että liiketoiminnallista vaikutusta. Riski voi olla teknisesti luokitukseltaan vakava, mutta jos se liiketoiminnallisesti ei ole merkittä, resurssien sijoittaminen kohteen turvaamiseksi voidaan nähdä turhana investointina [9 s.22][35 s.44].

Hyökkäys on toteutunut uhka ja se voi olla passiivinen tai aktiivinen. Passiivinen hyökkäys ei muuta viestin tai tiedoston sisältöä ja on vaikea havaita. Esimerkki tästä on salakuuntelu. Aktiivinen hyökkäys muokkaa tietoa, luo siihen uutta sisältöä tai häiritsee nykyisen tiedon käyttämistä [42]. Hyökkäysvektori kuvaa tapaa, jolla hyökkäys toteutetaan. Sähköpostin liitetiedosto, virus, saastuneet verkkosivut, ponnahdus ikkuna, pika-

viestimet tai suora kontakti palveluntarjoajan kanssa ovat hyökkäysvektoreita. Hyötykuorma kuvaa hyökkäyksen sisältöä. Esimerkiksi hyökkäysvektori voi olla sähköpostin liitetiedosto ja hyötykuorma liitetiedoston sisältämä virus [42].

3.1.3 Kohteet

Hyökkääjällä pitää Pfleegerin mukaan olla toteutustapa, mahdollisuus ja motiivi. Jos jokin näistä puuttuu, hyökkäystä ei tapahdu. Motiivi sisältää käsitteen kohde ja edellä mainittuihin ominaisuuksiin on vaikea vaikuttaa tietoturvapolitiikalla. Hyökkäyksen kannalta osaa kohteista voidaan kutsua kiinnostaviksi kohteiksi. Valtion laitokset, poliisi, armeija, pankit ovat kiinnostavia kohteita, koska ne sisältävät hyökkääjälle arvokasta informaatiota, tai koska niiden tietoturva on korkeammalla tasolla, on onnistunut hyökkäys suurempi haaste. Verkkohotellipalvelun markkinointi tietoturvallisena palveluna on hyvä myyntiargumentti, mutta samalla tämä laskee palvelun tietoturvaa. Hyökkääjät voivat nähdä tämän haasteena, jolloin palveluun murtaminen on entistä houkuttavampaa. Toinen yleinen kohde ovat helpot kohteet. Automatisoidut haavoittuvuusskannerit löytävät heikkouksia satunnaisista järjestelmistä ja raportoivat niistä hyökkääjille. Tietojen kerääjä ei aina ole sama henkilö kuin itse hyökkäyksen toteuttaja, ja raportteja löydetyistä haavoittuvuuksista myydään eteenpäin. Kolmas ryhmä on henkilökohtaiset kohteet. Hyökkäyksen motiivi voi olla poliittinen, uskonnollinen, taloudellinen tai henkilökohtainen [35 s.5;9-10;32].

Jotta riskit voidaan arvioida, täytyy kohde ja sen arvo määritellä. Yksi tapa jaotella kohteet on laitteisto, ohjelmistot ja tiedot. Laitteet ovat fyysisiä komponentteja ja niille sovelletaan tietoturvassa fyysistä koskemattomuutta, eli vain niihin oikeutetut henkilöt pääsevät niihin käsiksi. Laitteisiin ja niiden tietoturvaan vaikuttavat lisäksi luonnon katastrofit, sähkökatkot, pöly, lämpö, varastaminen ja inhimilliset virheet. Usein laitteistouhat, varsinkin palveluissa, on pienen hallintaryhmän vastuulla ja käsitellään yritystason tietoturvapolitiikassa [35 s.33].

Laitteistot tarvitsevat aina ohjelmistoja. Yksinkertaistettuna alimpana on käyttöjärjestelmä, joka on yhteydessä laitteistokomponentteihin. Käyttöjärjestelmän päälle asennetaan hyötysovelluksia, jotka keskustelevat käyttöjärjestelmän kanssa. Ohjelmia voidaan korvata, muokata, poistaa tai niitä voidaan käyttää luvattomasti. Ohjelman poistaminen voi olla osa hyökkäystä tai tapahtua vahingossa. Ohjelmapäivityksiä varten käytetään hallintatyökaluja, jotka hoitavat päivitykset määritettyjen protokollien mukaisesti ja ottavat varmuuskopion ohjelman tilasta ennen päivitystä. Tällöin ongelmatilanteissa voidaan palata nopeasti vanhaan toimivaan versioon. Ohjelmien päivittäminen on hyvin merkittävää tietoturvan kannalta, koska verkkoskannauksissa usein etsitään juuri ohjelmia, joiden päivityksen eivät ole ajan tasalla. Tietoturva on jatkuvana prosessi ja yksi tärkeimmistä vastatoimista on ohjelmien ja sovellusten jatkuva päivittäminen. Ohjelmilla on useita käyttäjäryhmiä, kuten ohjelmoijia, testaajia, ja käyttäjiä. Eri ryhmille asetettavat tietoturvavaatimukset vaihtelevat ja niiden ylläpito vaatii todentamista ja valtuuttamista [35 s.34].

Alkuperäisen ohjelmakoodin tulkitseminen vaatii erikoisosaamista, mutta se on erinomainen tietolähde takaporttien ja virheiden etsijöille. Ohjelmistoihin kohdistuneet hyökkäykset on helppo havaita jos ohjelma lopettaa toimintansa. Vakavampi uhka on ohjelmien modifiointi, jolloin ohjelma toimii alkuperäisellä tavalla, mutta tekee taustalla jotain muuta. Hyökkääjät ja ohjelmoijat voivat tehdä ohjelmiin muutoksia, jotka aktivoituvat vasta myöhemmin. Tällöin puhutaan loogisista pommeista. Erilaisia ohjelmamodifikaatioita on listattu alla [35 s.35-36].

- Troijalanen on ohjelma, joka näennäisesti tekee yhtä asiaa, mutta piilossa jotain muuta
- Virus on usein troijalainen, joka voi lisäksi siirtyä koneesta toiseen
- Takaportti tarkoittaa ohjelmassa olevaa piilotettua ominaisuutta tai haavoittuvuutta, joka esimerkiksi mahdollistaa laittoman kirjautumisen sovellukseen
- Tietovuoto tarkoittaa järjestelmässä olevan tiedon saatavuuteen liittyviä ongelmia
- Ohjelmistovarkaus tarkoittaa ohjelman lisenssiehtojen vastaista käyttöä

Järjestelmien sisältämä tieto on usein niiden arvokkain osa ja sitä voidaan saada haltuun monin tavoin. Tapoja ovat murtautumalla tietojärjestelmiin, kuuntelemalla verkkoliikennettä, kaivamalla roskakoreja, lahjomalla työntekijöitä joilla on oikeus tietoihin tai yksinkertaisimmillaan pyytämällä sitä tiedon omistajalta. Jos tietoja säilytetään selkokielisessä muodossa, on tiedon luotettavuus ja jäljitettävyys tärkeää pystyä takaamaan. Tiedon kryptaaminen tarkoittaa tiedon saattamista muotoon, josta vain salausavaimen tuntevat tahot voivat palauttaa sen selkokieliseen muotoon. Tiedon varastaminen, ostaminen tai löytäminen ei vaadi tietoteknistä osaamista, mutta tiedon muokkaamiseen tai uuden tiedon luomiseen järjestelmään vaatii tietoa siitä, miten tietoa tallennetaan, välitetään ja ylläpidetään. Tiedon kerääminen on usein hidas prosessi. Yksi tunnetuista ja vaikeasti havaittavista hyökkäyksistä on salami-hyökkäys. Siinä tietoa kerätään useilta eri käyttäjiltä, useista eri lähteistä pienissä osissa ja kasataan myöhemmin yhteen loogiseksi kokonaisuudeksi. Useat onnistuneet verkkohyökkäykset perustuvat pitkään kestäneeseen kokonaiskuvan muodostamiseen ja sen hyödyntämiseen [35 s.39].

Laitteilla ja ohjelmilla on arvo hankintahetkellä, mutta niistä tulee arvottomia ajan kuluessa ja tämän jälkeen niitä ei tarvitse suojata. Tiedon arvo arvioidaan tapauskohtaisesti. Se voi olla arvokasta vain hetken, tai se ei saa koskaan tulla julkiseksi. Esimerkiksi vaaleissa äänestyskuponkien sisältö täytyy suojata tuloksen julkaisemiseen asti. Tämän jälkeen tiedon suojaaminen ei ole tarpeellista. Jos tulosta on muokattu, niin tämän tiedon omistaja ei varmaan halua että tieto tulee koskaan julkiseksi [35 s.36-37].

Yrityksen avainhenkilöt ovat osa tietoturvallisuutta ja voivat olla hyökkäyksen kohteita. Tietoturvapolitiikan pitäisi käsitellä myös tilanteet, jolloin palvelulle tai tiedolle tärkeät avainhenkilöt irtisanoutuvat, loukkaantuvat tai muuten estyvät palvelun ylläpitotehtävistä. Samoin kuin tietoa säilytetään järjestelmissä, tieto voi olla sidottuna työntekijöiden osaamiseen ja tämän tiedon menettäminen tai vuotaminen kilpailijalle voidaan nähdä tietoturvariskinä. Henkilö jolla on oikeuksia tietoihin voi käyttää niitä väärin ja

näin aiheuttaa vahinkoa yritykselle. Avainhenkilöiden rekrytointi pitäisi nähdä osana tietoturvaprosessia.

3.1.4 Hyökkääjien profilointi ja heidän tavoitteet

Yleisesti hyökkääjät voidaan jakaa kolmeen ryhmään osaamisen tasolla. Osaamisen taso ei kerro tekijän motiiveista. Amatöörit ovat suurin ryhmä raportoiduissa tietoturvarikkeissä. Tämän ryhmän jäsenet löytävät vahingossa tai tahallisesti järjestelmistä tietoturva-aukkoja ja käyttävät niitä tilanteen tullen hyväkseen. Ryhmän jäsenet ovat usein taitavia tietokoneen käyttäjiä, jotka esimerkiksi työnsä ohella huomaavat, että heillä on tarkoitusta laajemmat käyttöoikeudet tai muu mahdollisuus päästä käsiksi arvokkaaseen tietoon. Jos tietoturvaongelmien raportointiin ei ole toimivaa järjestelmää, niin ongelmat harvoin tulevat ylläpidon tietoon. Vaikka tietoja ei heti käytettäisi hyväksi, niistä voi olla hyötyä esimerkiksi työpaikan vaihtamisen yhteydessä [35 s.41].

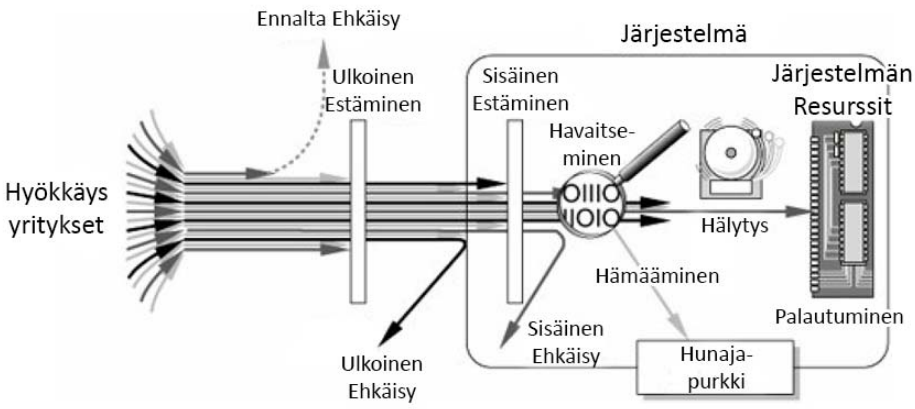
Hakkerit ja krakkeri ovat usein hyvin päteviä tietotekniikan osaajia, jotka yrittävät päästä käsiksi järjestelmiin ja tietoihin joihin heillä ei ole oikeuksia. He usein näkevät tietomurrot uhrittomina rikoksina ja haasteina. Motiivi tietomurroille voi olla yleinen kiinnostus, taloudellinen hyöty, henkilökohtaiset haasteet tai harmin aiheuttaminen. Tietoturvayhteisö pitää hakkerin ja krakkerin erona motiivia. Hakkeri työskentelee järjestelmien parissa niiden testaus- ja kehitysmielessä, sekä ymmärtääkseen niiden toimintaa. Krakkerin motiivi on vihamielinen ja tavoitteena henkilökohtainen etu. Tilanne ei ole näin mustavalkoinen ja myös sanaa hakkeri käytetään negatiivisessa merkityksessä. Lisäksi verkkosovellusten ja tietoturvajärjestelmien ympärille on kehittynyt ammattirikollisten ryhmä. Rikolliset eivät vaihda alaa autovarkauksista tietoturvarikoksiin, vain ovat usein tietotekniikan alalla työskenteleviä ammattilaisia, joiden motiivi on pääasiassa raha. Ammattimaiset tietomurrot ja virukset liittyvät usein hallituksien toimintaan, politiikkaan ja terrorismiin [35 s.42-43].

3.1.5 Vastatoimet

Vastatoimet kuvaavat miten tietoturva toimii hyökkäyksiä vastaan. Alla vastatoimet on jaoteltu Pfleeganin 2009 mukaan seuraavasti:

- **Ehkäisy** tarkoittaa hyökkäyksen torjumista ja haavoittuvuuden poistamista
- **Estäminen** tarkoittaa hyökkäyksen toteuttamisen vaikeuttamista
- **Hämääminen** tarkoittaa, että kohteesta tehdään ei-mielenkiintoinen, tai toisista kohteista tehdään mielenkiintoisempia
- **Havaitseminen** voidaan tehdä hyökkäyksen aikana tai sen jälkeen
- **Palautuminen** tarkoittaa prosessia, miten hyökkäyksestä selvitään ja palaudutaan hyökkäystä edeltävään tilaan
- **Hälytys** tarkoittaa, että järjestelmän valvoja saa tietää, jos jokin uhkakuva toteutuu ja tähän voidaan tällöin reagoida mahdollisimman nopeasti

Suojautuminen on näiden toimien summa. Kuvassa 3.2. on esitetty järjestelmä ja miten eri vastatoimet kerrostetaan yhtenäiseksi kokonaisuudeksi.



Kuva 3.2. **Vastatoimien mahdollinen rakenne [9 s.46]**

Kuvasta huomataan, että ensimmäinen tapa suojautua hyökkäyksiä vastaan on ennaltaehkäisy. Tämä voi tarkoittaa esimerkiksi ilmoitusta siitä, että yrityksellä on tietoturvaohjelma käytössä ja kaikki hyökkäysyritykset käsitellään ja ilmoitetaan tarvittaessa poliisille. Hyökkääjän henkilökohtaisen riskin nostaminen on tehokas tapa torjua hyökkäys jo ennen sen alkamista [35 s.44].

Tietoturvallisuuden ja hallintajärjestelmien sertifiointi tarkoittaa todistusta tietoturvallisista toimintatavoista. International Organization for Standardization (ISO) on määritellyt kriteerit tarvittavia toimenpiteitä varten ISO/IEC 27001-standardissa. Viralliset standardit ovat maksullisia, mutta verkosta on saatavilla vastaavia ilmaisia teoksia, kuten ISF:n laatima The Standard of Good Practise for Information Security(julkaistu 2007). Suomenkielisiä ohjeita julkaisee Valtiohallinnon VAHTI-työryhmä. Tietoturvakartoituksen yhteydessä puhutaan usein lisäksi tietoturvan auditoinnista. Auditointi vertaa yrityksen toimintaa olemassa olevaan tietoturvapolitiikkaan, mutta ei itsessään kehitä toimintaa tai ota kantaa, jos itse tietoturva politiikka on vanhentunutta [7 s.16].

3.1.6 Tietoturvan tavoitetila

Tietoturvan tavoitetila on tilanne, jossa mahdolliset hyökkäysvektorit on kartoitettu, niihin on varauduttu, haavoittuvuuksien testaaminen on automatisoitu ja kaikki tiedon parissa työskentelevät henkilöt on koulutettu toimimaan eri tilanteissa. Tämä ei ole mahdollista, mutta tavoitetilan määrittämisen avulla on mahdollista tehdä suunnitelma ja asettaa tavoitteet tietoturvapolitiikalle. Tietoturvallisuuden johtaminen on prosessi, jonka tarkoitus on arvioida tiedon arvoa yritykselle, tunnistaa siihen liittyvät riskit sekä varautua niihin eri keinoin. Prosessi sisältää jatkuvan uhkien ja vastatoimien arvioinnin, sekä prosessien kehittämisen näiden arvioiden perusteella. Tämän prosessin ohjausta kutsutaan tietoturvapolitiikaksi. Tietoturvapolitiikka kattaa yrityksen yleiset toimintatavat ja sen alle voidaan luoda prosessikohtaisia tietoturvapolitiikkoja [43].

4 VERKKOSOVELLUSTEN TIETOTURVA JA YLEISET UHAT

Tässä luvussa käsitellään eri lähteistä kerätyn tiedon pohjalta yleisimmät verkkosovelluksiin kohdistuvat uhat ja miten niitä vastaan voidaan suojautua. Uusia hyökkäyksiä kehitetään jatkuvasti ja tietoturvaohjelmistot ovat usein askeleen jäljessä kehityksessä. Osa tietoturvasta ja sen markkinoinnista perustuu aina pelkokuvien luomisesta. Vaikka ongelmia on olemassa, niin täytyy muistaa, että oikein ohjelmoidun ja päivitetyn verkkosovelluksen joutuminen onnistuneen hyökkäyksen kohteeksi on kohtuullisen pieni.

Luvussa kolme on todettu, että tieto itsessään on usein palvelun arvokkain ominaisuus ja palveluita testattaessa täytyy määritellä, kuka käsiteltävän tiedon omistaa. Tässä työssä käsiteltävän palvelun tuottamiseen käytetyt fyysiset palvelimet, joilla tieto sijaitsee, on Decens Oy:n omistuksessa, mutta itse tieto on asiakkaan omistamaa. Jos asiakkaan omistamaa tietoa muutetaan tai luetaan luvatta, rikotaan asiakkaan tietoturvaa. Palveluntarjoajan vastuulla on verkkohotellipalvelun alustan toimivuus ja se ei ole vastuussa asiakkaiden verkkosivujen toteutuksesta. Silti palveluntarjoajan on oltava tietoinen palvelimillaan tapahtuvista prosesseista, pystyä puuttumaan niihin tarvittaessa ja tarjota tukea ongelmatilanteissa. Ennakoivalla ongelmien kartoittamisella voidaan ehkäistä hyökkäysten onnistuminen ja tämä voidaan käsittää palvelun laadun parantumisena.

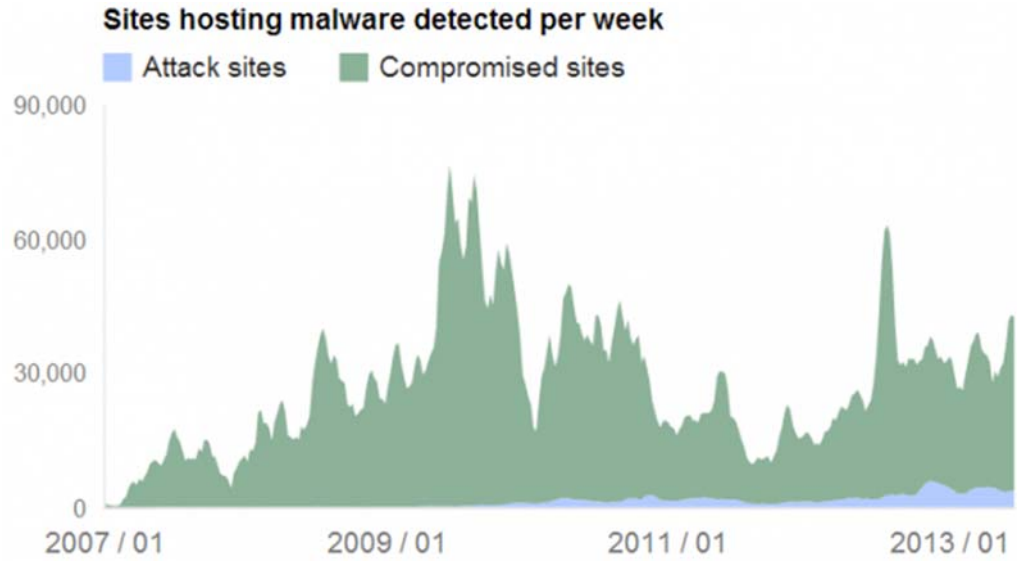
4.1 Palveluntarjoajaan ja asiakkaisiin kohdistuvat uhat

Verkkohotellipalveluun kohdistuvat uhat voidaan karkeasti jakaa palveluntarjoajaan tai asiakkaisiin kohdistuneisiin uhkiin. Palveluntarjoajaan kohdistuvat uhat tarkoittavat palvelun toteuttamiseen tarvittavien komponenttien, kuten palomuurien, palvelimien ja ohjelmistojen haavoittuvuuksiin kohdistuviin hyökkäyksiin. dHosting-palvelu on ostanut palvelun toteuttamiseen käytetyt ohjelmat kolmannelta osapuolelta ja niiden toimittaja vastaa ohjelmien tietoturvanpäivitysten jakelusta. Fyysisiä komponentteja koskeva tietoturva on määritelty yrityksen yleisessä tietoturvapolitiikassa ja sen on todettu riittäväksi myös verkkohotellipalvelun toteuttamiseen. Lisäksi palveluntarjoaja vastaa, että tieto ei katoa lopullisesti palvelusta ja se ylläpitää varmuuskopiota palvelusta. Varmuuskopiosta palauttaminen viiden edellisen vuorokauden ajan palautuspisteistä onnistuu lisäksi asiakkaan hallintapaneelista.

Asiakkaan vastuulla on palvelun sisällön ylläpito. Palvelusopimuksessa asiakkaalle tarjotaan rajat kotisivutilan ja verkkoliikenteen käyttöön. Asiakas vastaa itse verkkosovellustensa tietoturvasta, asiakastiedoista ja sisällöstä. Nämä ovat tietoja, joihin palveluntarjoaja ei yleensä pääse edes käsiksi ja usein arkaluotoiset tiedot säilytetään kryptatussa muodossa. Palveluntarjoajan on hyvä seurata millaisia verkkosivuja ja -sovelluksia asiakkaat ajavat kotisivutiloissaan. Laki velvoittaa palveluntarjoajaa puuttumaan asiakkaiden sivujen sisältöön, jos se on lainvastaista (kiihottaminen kansanryhmää vastaan,

loukkaavan kuvan levittäminen tai sukupuolisiveellisyyttä loukkaava materiaali) tai sivujen sisältö rikkoo tekijänoikeuksia. Palveluntarjoajan tulee estää tällaisen sisällön saanti saatuaan asiasta oikeuden määräyksen tai omasta aloitteestaan saatuaan tiedon sisällön olemassaolosta. Tällaisen materiaalin poistaminen ei ole pakollista, vaan tiedon saanti täytyy ainoastaan estää. Tiedon poistaminen ei edes ole suotavaa mahdollista rikosprosessia varten [45].

Usein asiakas ei ole edes tietoinen sivuillaan sijaitsevasta haitallisesta materiaalista. Kuvassa 4.1. on Googlen tutkimus verkossa havaituista haitallisista sivuista.

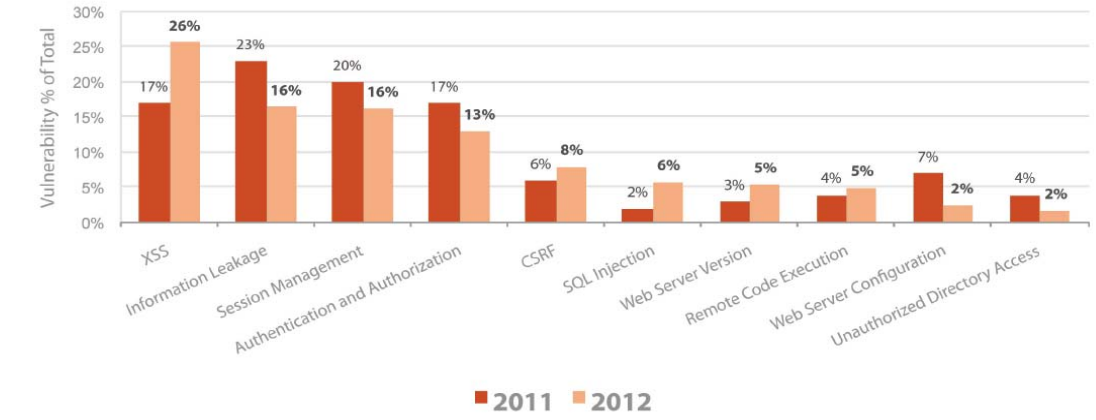


Kuva 4.1 **Googlen mittaus saastuneiden sivujen määristä [46]**

Tutkimuksesta huomataan, että kaapatuille sivuille ujutetut haittaohjelmat ovat yleisempi uhka, kuin itse vihamieliset sivustot. Hyökkääjien on tehokkaampaa ja edullisempaa etsiä verkkosivuilla olevia haavoittuvuuksia, kuin tuottaa uusia sivustoja. Tällöin hyökkääjän ei tarvitse houkutella kävijöitä sivuilleen, vaan hyökkäyksen kattavuus riippuu vaarantuneiden sivujen suosiosta. Tekniikka on erityisen hankala tietoturvaohjelmille, jotka perustuvat sivujen maineeseen [46].

4.2 Yleisimmät uhat

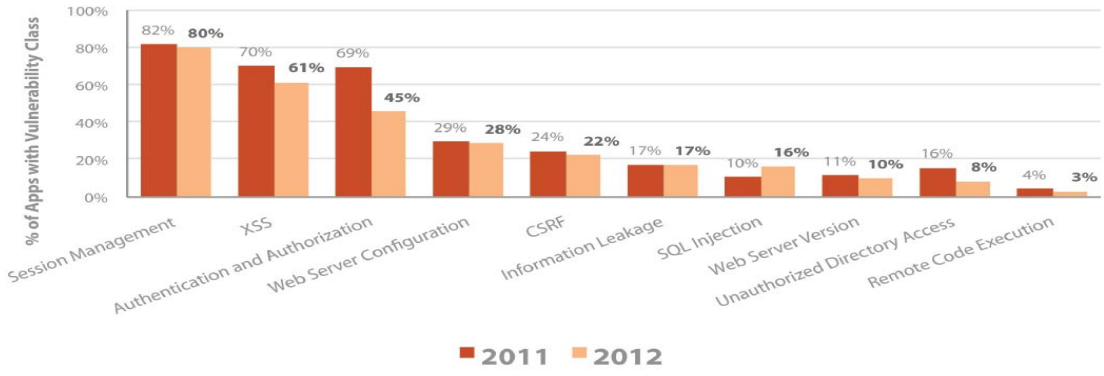
Tietoturvayritykset analysoivat verkkopalveluita ja tarjoavat tilastotietoa verkossa tapahtuvista hyökkäyksistä. Tietoturvayhtiön Cenzicin vuonna 2012 julkaistussa raportissa ilmoitetaan, että 99% verkkosovelluksista sisältää haavoittuvuuksia ja keskimäärin sovelluksessa oli 13 haavoittuvuutta. Löydettyjen haavoittuvuuksien jakauma vuosilta 2011 ja 2012 on esitetty kuvassa 4.2.



Kuva 4.2 Eri verkkohyökkäysten tilastollinen jakauma vuodelta 2011 ja 2012 [47]

Tilastosta nähdään, että vuoden 2012 yleisin uhka on ollut XSS, eli Cross Site Scripting. Positive Technologiesin, vastaavassa tutkimuksessa vuonna 2011 XSS-hyökkäys sai 21 prosenttia ja tietojen vuotaminen 17 prosenttia tuloksista. Tulokset ovat samansuuntaisia, vaikka tutkimusten mittarit eivät ole suoraan verrattavissa.

Kuvassa 4.3. Cenzicin mittaus on esitetty haavoittuvuuskohtaisesti eri sivuilla.



Kuva 4.3 Sivukohtaisten haavoittuvuuksien esiintyminen vuosilta 2011 ja 2012

Vuonna 2012 80 prosenttia testatuista sivuista kärsi istunnonhallinnassa olevista haavoittuvuuksista ja 61 prosenttia sisälsi XSS-haavoittuvuuksia. Kyseessä on tietoturvayritysten julkaisemista mittauksista, joten niiden puolueellisuus on kyseenalaista. Tilaston tieteellistä arvoa laskee myös se, että otoksen kokoa ei ole ilmoitettu. Tuloksista saadaan kuitenkin suuntaa-antava näkymä tämän hetken haavoittuvuuksiin ja niitä voidaan käyttää apuna testaussuunnitelman laatimisessa [47][48].

OWASP (Open Web Application Security Project) on vuonna 2001 perustettu, voittoa tavoittelematon järjestö, jonka tavoite on ohjelmistojen tietoturvan parantaminen. Järjestö julkaisee vuosittain listan kymmenestä yleisimmästä sovelluksia koskevista tietoturvariskeistä [9 s.6]. Vuoden 2013 tulokset vastaavan tietoturvayritysten esittämiä tuloksia ja ne on esitetty taulukossa 4.1.

Taulukko 4.1 OWASP-järjestön kymmenen vaarallisinta uhkaa verkkosovelluksille vuonna 2013

Vaarallisuus	Uhka
1	Injections
2	Broken authentication and session management
3	Cross-Site Scriptin
4	Insecure Direct Object Referenses
5	Security Misconfiguration
6	Sensitive Data Exposure
7	Missing Function Level Access Control
8	Cross-Site Request Forgery (CSRF)
9	Using Components with Known Vulnerabilities
10	Unvalidated Redirects and Forwards

Taulukosta huomataan, että erilaiset injektiot, haavoittuvuudet kirjautumisessa ja istunnonhallinnassa, XSS ja tietojen vaarantuminen ovat yleisimpiä haavoittuvuuksia. Osa listan uhkista kohdistuu palvelinasetuksiin, tai verkkosivuilla sijaitsevan tiedon hallintaa.

4.2.1 Injenktiot

Ensimmäisenä OWASP-listalla on Injektiot. Injektio on hyökkäys, jossa hyökkääjä välittää ohjelmakoodia sovelluksesta toiseen. Injektiohyökkäys voi olla järjestelmäkomentoja käyttävä käyttöjärjestelmäinjektio, tietokantainjektio tai esimerkiksi tunnistautumiseen liittyvä injektio. Injektion sisältö voi olla tekstiä, numeroita, HTTP-pyyntöjä tai jopa kokonaisia eri ohjelmakielillä kirjoitettuja ohjelmistoja, jotka injektoiva kohde saadaan pahimmassa tapauksessa suorittamaan [50]. SQL-injektio on yksi tunnetuimmista injektioista. Se on haavoittuvuus, jossa hyökkääjä antaa palvelun taustalla toimivalle tietokantasovellukselle muokattuja pyyntöjä. Jos pyyntö on oikein muodostettu, hyökkääjä voi saada haltuunsa, muokata tai poistaa tietoja tietokannasta [9 s.6-7].

Injektiouhkien löytämiseksi käytetään verkkosivuja analysoivia ohjelmia, jotka etsivät ja tarkistavat sovelluksille tehtäviä pyyntöjä ja komentoja. Tällaisia kohteita ovat esimerkiksi SQL-pyynnöt, käyttöjärjestelmä komennot, tai eri ohjelmointikielien tulkeille lähetetyt komennot [9 s.7]. Analysointiohjelmien tulokset eivät ole täysin luotettavia ja on suositeltavaa, että löydetyt haavoittuvuudet testataan lisäksi käytännössä. Tietokantakyselyjen on suositeltavaa määritellä täyttämään ennalta annetut parametrit. Esimerkiksi syöte käyttäjänimelle voidaan rajata maksimissaan 20 merkkiä pitkäksi ja se saa sisältää vain kirjaimia ja numeroita. Toinen tapa suojautua injektioilta on antaa verkko-

sovelluksille vain ne oikeudet, jotka se ehdottomasti vaatii. Verkkosovelluksen ei ole suositeltavaa koskaan toimia verkkopalvelimen tai tietokantasovelluksen pääkäyttäjän tunnuksilla [50]. Injektiohyökkäyksen seurauksia ovat tiedon luottamuksellisuuden ja eheyden menettäminen. Lisäksi tietojen poistaminen voi johtaa saatavuuden menettämiseen. Usein palveluiden käyttäjätunnukset tallennetaan tietokantaan. Onnistuneen injektion avulla on mahdollista lisätä uusia tunnuksia järjestelmään, kirjautua toisena käyttäjänä, tai lisätä jo olemassa olevan käyttäjän oikeuksia, ja näin päästä käsiksi salattuun tietoon [51].

SQL-tietokantakielen syntaksi on selkokielistä ja esimerkki SQL-injektiosta on esitetty esimerkissä 4.1.

```
SELECT * FROM users WHERE user_name = '" + user_name + "';        (4.1)
```

Esimerkissä verkkosovellus suorittaa tietokantakutsun, johon generoidaan mukaan käyttäjän nimeä kuvaava user_name-muuttujasta. Jos käyttäjä antaa nimekseen tässä tilanteessa "1; DROP TABLE users; --"-merkkijonon, on tietokannalle lähetettävä pyyntö muotoa, jossa SELECT-komennon lisäksi suorittaisiin DROP-komento. Tämä tilanne on kuvattu esimerkissä 4.2.

```
SELECT * FROM users WHERE user_name = ''; DROP TABLE users; --';  (4.2)
```

Esimerkissä tietokantakutsu suorittaa SELECT-komennon puolipisteeseen asti, jonka jälkeen suoritetaan DROP TABLE-komento. Tämä komento poistaa tietokannasta ”users”-nimisen taulun. Kaksi väliviivaa merkitsevät SQL-kielessä kommenttia, ja tällöin loppuriviä ei enää suoritettaisi. Tämä on esimerkissä turha, mutta jos komennossa olisi muita käskyjä, voisi kääntäjä tulkita rivin virheelliseksi ja palauttaa virheilmoituksen. Tällaiset hyökkäykset voidaan estää kieltämällä heittomerkin tai kommentointien käyttö syötteissä [6 s.109-111].

SQL-injektioiden haavoittuvuuksien kartoittamiseen on kolme eri tekniikkaan. Virhepohjainen SQL-injektio perustuu tietokannalla lähetettyyn virheelliseen pyyntöön, johon tietokanta vastaa tavalla josta haavoittuvuus voidaan päätellä. Yhdistelmähyökkäys tarkoittaa useasta laillisesta pyynnöstä kasattua tietoa. Kolmas vaihtoehto on sokea SQL-injektio. Kaikki SQL-injektiot eivät anna hyökkääjälle näkyvää palautetta hyökkäyksen onnistumisesta, vaan hyökkääjä joutuu kokeilemaan useita vaihtoehtoisia syötteitä. Esimerkiksi jos tietokanta antaa jokaiseen virhepohjaiseen SQL-hyökkäykseen saman virheraportin, niin sille voidaan antaa pyyntö muodossa: ”Jos käyttäjänimi ’admin’ on olemassa, odota 10 sekuntia.”. Jos virheilmoitus ilmestyy vasta 10 sekunnin kulutta, voidaan tästä päätellä että kyseinen käyttäjätunnus on olemassa. Hyökkäysohjelmat ovat hyvin kehittyneitä ja voivat tunnistaa onko testattava käyttäjäntunnus olemassa tietokantapalvelimen vastausajan perusteella. Ei olemassa olevaa käyttäjätunnusta varten täytyy käydä

läpi koko tietokanta, toisin kuin vain väärä salasana selviää heti oikean käyttäjätunnuksen löydyttyä [6 s.117-121].

4.2.2 Kirjautuminen ja istunnonhallinta

Todentaminen (engl. Authentication)ja valtuuttaminen (engl. authorization) liittyvät palvelun istunnonhallintaan ja palvelun käyttöön. Verkkopalveluihin tunnistautuminen ja istunnon ylläpito ovat vahvasti sidoksissa. Jos jompikumpi hoidetaan huonosti, niin toisestakaan ei ole hyötyä [6 s.144]. Todentaminen tarkoittaa käyttäjän tai palveluntarjoajan tunnistamista. Salasana on yleisin tapa suorittaa käyttäjän todentaminen ja tätä pidetään yleisesti luotettavana tekniikkana. Palveluntarjoaja käyttää SSL-sertifikaattia todistaakseen kuka on ja että verkkosovellus toimii väittämässään verkko-osoitteessa. Valtuuttaminen tarkoittaa todennetun käyttäjän oikeuksien rajoittamista nimettyihin tietoihin tai toimintoihin. Esimerkiksi verkkopankin ensimmäinen vaihe on todentaa käyttäjä ja tämän jälkeen valtuuttaa käyttäjä käsittelemään vain omia tilejään. Verkkosovellusten toteuttamiseen käytetyistä ohjelmointikielistä, kuten PHP ja ASP, on valmiiksi rakennettu istunnon luomiseen tarvittavat komponentit. Samoin yleisimmistä julkaisujärjestelmistä löytyvät nämä ominaisuudet valmiina. Todentamista vastaan voidaan hyökätä kalastelemalla käyttäjän salasana, tai ohittamalla jollain tekniikalla koko todentamisprosessi [6 s.141-142]. Yleisiä ongelmia koskien kirjautumista ovat liian lyhyet, toistuvat ja yleiset salasanat, salasanojen tallentaminen selaimeen, sekä huonosti hoidettu uloskirjautuminen verkkosovelluksista. Palveluntarjoaja huolehtii, että käyttäjätunnukset ja salasanat tallennetaan turvallisesti kryptaamalla tai käyttäen tiivistettä [54][9 s.8].
PC World on analysoinut artikkelissaan 2011 Anonymous-hakkeriryhmän julkaisemien käyttäjätunnus-salasana -parien pohjalta salasanojen toistuvaa käyttöä kahdessa eri verkkosovelluksessa. Tuloksien mukaan 49 prosenttia molemmissa palveluissa olevista käyttäjistä käytti samaa salasanaa molemmissa palveluissa ja kuusi prosenttia käytti samaa salasanaa muokaten sitä hieman. Aikaisemmin samankaltaisen tutkimuksen tulos on ollut 12-20 prosentin luokkaa [55]. Verkkosovelluksen turvatoimilla ei ole merkitystä, jos hyökkääjä saa käyttäjätunnuksen ja salasanan käyttöönsä toisen palvelun kautta. Verkkopalveluiden kirjautumista vastaan voidaan hyökätä myös automatisoidulla ohjelmilla. Brute Force-tekniikka arvaa eri käyttäjätunnusten ja salasanojen yhdistelmiä. Tekniikka on yksinkertainen ja tehokas, koska kyse on vain ajasta ja laskentatehosta. Testattavien kombinaatioiden määrä on huomattavasti pienempi, jos hyökkääjällä on olemassa oleva käyttäjätunnus valmiina tiedossa. Jos käyttäjätunnusta ei ole olemassa, niin sille ei kannata yrittää arvata salasanaa. Verkkosovellukset ja sisällönhallintajärjestelmät usein luovat ylläpitäjän käyttäjätunnuksen asennuksen yhteydessä. Tunnus on usein oletuksena muotoa ”admin” tai ”root” ja järjestelmän luoman tunnuksen vaihtaminen on suositeltavaa. Kirjautumisen yhteydessä tapahtuvat virheilmoitukset eivät saa vaihdella eri tilanteissa. Jos sovellus antaa ilmoituksen, että yritetty käyttäjätunnus on olemassa, mutta salasana on väärä, saa hyökkääjä helposti tarvitsemaansa tietoa palvelusta. Tällainen haavoittuvuus on teknisesti virhepohjainen SQL-injektio. Tästä huomataan, että kaikkien uhkien määrittäminen eri kategorioihin ei ole yksiselitteistä [56][6 s.146,165].

Valtuuttaminen liittyy käyttäjätunnusten luomiseen ja hallintaan. Valtuuttamiseen liittyvää prosessia suunniteltaessa päätetään kuka voi lisätä itsensä palveluun käyttäjäksi, onko prosessi automatisoitu, tehdäänkö kirjautumiseen sähköpostitarkistus, voiko sama henkilö rekisteröityä useasti ja millaisen roolin käyttäjä saa sovellukseen. Nykyisissä verkkopalveluissa oletetaan olevan vähintään kahden tason tunnuksia: käyttäjiä ja ylläpitäjiä. Käyttäjätaso ei aina vaadi kirjautumista. Kirjautumista tutkivat tietoturvaohjelmat ottavat kantaa teknisiin virheisiin ja tutkivat palvelimien konfiguraatiotiedostoja, kuten esimerkiksi Apache-verkkopalvelimen .htaccess-tiedostoa. Prosessi on vaikea automatisoida ja ohjelmat antavat usein virheellisiä positiivisia varoituksia [56]. Oikeudet määritellään sovelluskohtaisesti, jolloin tehokas työkalu on tehdä oikeuksista matriisi ja käydä eri tilanteet läpi manuaalisesti. Taulukossa 4.2 on esitetty esimerkki tällaisesta matriisista.

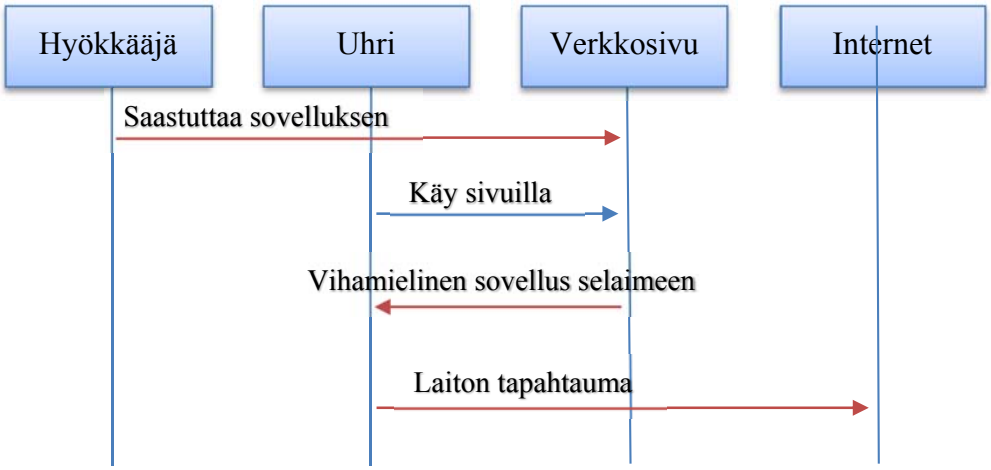
Taulukko 4.2 Esimerkki valtuuttamiseen liittyvistä rooleista

Rooli	Oikeudet	Rajoitukset
Ylläpitäjä	Luku ja kirjoitus	Ei rajoituksia
Käyttäjä	Luku ja kirjoitus	Vain itseään koskevat tiedot
Ohjelmoija	Luku	Erikseen luodut testitunnukset

Kun oikeudet on määritelty, voidaan palveluun kirjautua eri tason tunnuksilla ja analysoida verkkosivujen sisältö. Käyttäjälle esitettävät resurssit voidaan kartoittaa tarkoitukseen sopivalla ohjelmalla, kuten Screaming Frog tai Spider Test Tools [57].

4.2.3 Cross Site Scripting

Cross Site Scriptin, eli XSS, perustuu hyvämaineisella sivulla sijaitsevaan ohjelmointivirheeseen, joka mahdollistaa vihamieleisen ohjelmakoodin asettamiseen sivuille. Yleisempiä hyökkäyksen kohteita ovat HTML, JavaScript, VBScript, ActiveX ja FLASH. Tarkoitus on muuntaa ja väärentää selainohjelmassa ajettavia ohjelmakoodeja niin, että sivusto suorittaa hyökkääjän haluamia toimintoja. Kun käyttäjä menee sivuille, selain lataa vihamielisen ohjelmakoodin ja se suoritetaan osana sivun laillista ohjelmakoodia. Hyökkäyksestä on useita variaatioita ja kuvassa 4.4 olevassa diagrammissa on kuvattu pysyvän XSS-hyökkäyksen rakenne [58].



Kuva 4.4 **Esimerkki Pysyvän XSS-hyökkäyksen rakenteesta**

Esimerkissä krakkeri syöttää verkkosivulle vihamielisen ohjelmakoodin, jonka käyttäjän selain suorittaa vierailun yhteydessä. Huomattavaa on, että hyökkäyksen ohjelmakoodi ajetaan käyttäjän selaimessa, ei palvelimella. HTTP-protokolla välittää tietoa käytössä olevasta järjestelmästä, kuten esimerkiksi selainohjelman mallin ja version. Hyvin tehty XSS-hyökkäys ei aktivoidu jokaisen käyttäjän kohdalla, vaan se nuuskii käyttäjän järjestelmää ja hyökkäys suoritetaan vain, jos kyseessä on sopiva kohde. Tämä vaikeuttaa hyökkäyksen havaitsemista, koska hyökkäys tapahtuu vain määrätyissä tilanteissa. XSS-hyökkäyksiä käytetään identiteettivarkauksiin, suojatun tiedon haltuun saamiseen, palveluiden maksujen kiertämiseen, käyttäjien vakoiluun sekä palvelunestohyökkäyksiin. Sovelluskehittäjät vähättelevät XSS-hyökkäyksiä, koska se avulla ei voi kaapata tietoja verkkosovelluksen taustatietokannoista. XSS-hyökkäyksen avulla hyökkääjä voi kuitenkin saada haltuunsa käyttäjätietoja, joiden avulla voidaan suorittaa luvattomia tietokantahakuja [58].

Sivujen ylläpitäjät voivat ajattelevat, että heidän sivustolleen ei voi lisätä ajettavaa ohjelmakoodia ja ovat tämän takia suojassa XSS-hyökkäyksiltä. XSS-hyökkäysvektoreita on useita ja monet verkkosovellukset lataavat ohjelmistokirjastoja ja sisältöä eri lähteistä. Esimerkiksi kuvan lataaminen linkkinä toiselta sivulta altistaa sovelluksen kuvatietoihin piilotetulle metadatalle. Verkkosovelluksissa on usein mahdollisuus jättää kommentteja. Jos kommenttikentän syötteitä ei käsitellä oikein, voidaan niihin piilottaa ohjelmakoodia. Kun kävijä lukee kommentin, niin tulee hän samalla mahdollisesti suorittaneeksi vihamielisen ohjelmakoodin. Suosituille sivuille ujutetusta vihamielisestä ohjelmakoodista käytetään termiä vesipaikka-hyökkäys. Nimi viittaa tapaan, jolla leijonat vaanivat vesipaikkojen lähistöllä saaliseläinten saapumista [58][59].

XSS-hyökkäysvektorit jaetaan heijastaviin ja jatkuviin hyökkäyksiin. Heijastavassa hyökkäyksessä hyökkääjä lähettää uhrille esimerkiksi sähköpostin, jossa on linkki vihamieliselle sivustolle. Houkuttelevasti muotoiltu linkki lähettää HTTP-pyynnön hyökkääjän palvelimelle, joka palauttaa uhrin koneelle vihamielisen resurssin. Pysyvissä

hyökkäyksissä hyökkääjä on löytänyt haavoittuvuuden verkkopalvelusta ja saanut tätä kautta asennettua verkkosivuille ohjelmakoodia, jonka käyttäjän selain suorittaa. Onnistuneen hyökkäysvektorin hyötykuorma voi olla esimerkiksi käyttäjän session evästeiden varastaminen tai näppäinlyöntejä tallentava ohjelma. Pahimmissa tapauksissa hyökkääjä saa uhrin koneen tai selaimen hallintaansa ja pystyy käyttämään hänen tunnistetietojaan eri palveluissa [60].

Useille sivuille myydään mainoksia. Jos mainosalueet eivät ole turvallisesti toteutettuja, voi kuka tahansa voi ostaa verkkosivuille ikkunan, jossa ajaa omaa ohjelmakoodiaan. Tällöin puhutaan termistä malvertising. On raportoitu tapauksia, joissa hyökkääjät ovat maksaneet sivujen ylläpitäjiltä sivuille tehdyistä muutoksista. [61 s.7] XSS-hyökkäysten yhteydessä puhutaan myös termistä drive-by downloadin (DbD). Käyttäjän vierailu verkkosivuilla aloittaa ohjelmakoodin lataamisen käyttäjän tietokoneelle. Oletuksena verkkosivun eivät voi ladata ohjelmakoodia käyttäjän koneelle ja suorittaa sitä. Tällöin hyökkäys perustuu ohjelmavirheeseen esimerkiksi selaimessa. Lisäksi ohjelmat kuten Java, Flash, Shockwave, Adobe Reader, Quicktime ja selaimiin asennettavat lisäosat mahdollistavat sivuilla olevan ohjelmakoodin suorittamisen ja ovat potentiaalisia kohteita verkkosivujen kautta tapahtuville hyökkäyksille [58].

Sovelluskehittäjät voivat suojautua XSS-hyökkäyksiä vastaan käyttämällä valmiita testauslistoja, joita verrataan sivuston ohjelmakoodiin. Testattavia kohteita on satoja ja näitä varten on tehty automatisoituja testausohjelmia. Testausohjelmat tulevat kuitenkin aina jäljessä, kun uusia haavoittuvuuksia keksitään. Verkkohotellipalveluissa yleisesti asiakas vastaa palvelulla sijaitsevasta tietosisällöstä ja haavoittuvuuksien testaaminen on sivuston ohjelmoijan vastuulla. Vaikka sovellus ei sisällä salaista tai suojausta tarvitsevaa tietoa, on ylläpitäjä vastuussa käyttäjien tietoturvasta. Käyttäjät voivat suojautua XSS-hyökkäyksiä vastaan selaimiin asennettavilla laajennuksilla, joiden avulla ohjelmakoodien toimintaa voidaan rajoittaa tai ne voidaan kytkeä kokonaan pois päältä [58].

4.2.4 Tietojen vuotaminen ja sovellusten konfigurointi

Verkkosovellukset sisältävät arvokasta tietoa ylläpidolle ja käyttäjille. Tietovuoto sovelluksessa voi vaarantaa tietoturvan myös muissa käyttäjän käyttämissä sovelluksissa, kuten huomattiin toistuvien salasanojen esimerkissä. Sovellukset sisältävät sähköpostiosoitteita, asiakastietoja, liiketoiminnallista tietoa, tai muita ei julkiseksi luokiteltuja dokumentteja. Kuten verkkohotellipalvelun ja sen asiakkaiden välillä, sovelluksen ylläpitäjän täytyy tietää kuka sovelluksessa olevan tiedon omistaa. Esimerkiksi Facebook joutui vuonna 2009 kohun keskelle käyttäessään käyttäjien yksityisiä kuvia mainoksissaan [6 s.201-202].

Neljäntenä uhkana OWASP-listalla on suorat kohdeviittaukset. Hyökkäys perustuu URL-pyynnön muokkaamiseen, jolloin hyökkääjä saa salaisen resurssin haltuunsa muuntamalla lähetetyn pyynnön rakennetta. Hyökkäys voidaan mieltää injektioksi, jossa käytetään hyväksi valtuutustarkistuksen puuttumista. Verkkosivujen rakenteesta voi löytyä suoria viittauksia salaiseen resurssiin. Esimerkiksi esityskerrokselle on päätynyt tie-

tokantapalvelimen osoite tai tietokantojen nimiä, vaikka nämä pitäisi olla piilossa logiikkakerroksella. Koska keskustelu resurssien ja logiikkakerroksen välillä ei pitäisi päätyä esityskerrokselle, tunnistautumiseen tarvittavat prosessit eivät ole yhtä tiukkoja kuin esityskerroksella [9 s.10].

Viidentenä uhkana listalla on tietoturvaan liittyvien asetusten huolimaton konfigurointi. Tämä virhe voi tapahtua millä tahansa tasolla, kuten palvelimella, sovelluksessa, taustajärjestelmissä, tietokantasovelluksessa, viitekehyksessä tai itse tuotetussa ohjelmakoodissa. Ongelma voi olla vanhentunut järjestelmä, heikko salasana, avoin verkkoportti tai ohjelmointivirhe. Yksinkertainen esimerkki on tilanne, jossa verkkosovelluksen alustaksi on asennettu julkaisujärjestelmä, mutta unohdettu vaihtaa oletuksen luotu ylläpidon käyttäjätunnus ja salasana. Kun hyökkääjä kartoittaa kohdesovellusta, voidaan tällaisten haavoittuvuuksien havaitseminen helposti automatisoida ja testata. Listan kuudes uhka on arkaluontoisen tiedon paljastuminen. Uhka syntyy pääasiassa siitä, että salaista tietoa ei salata oikein, tai sitä ei ole määritelty salaiseksi tiedoksi. Usein hyökkääjät eivät murra itse salausta, vaan yrittävät kaapata tiedon avaamiseen tarvittavan salausavaimen. Ei-salattua tietoa voidaan kaapata lisäksi tietoliikenteen seasta ja selaimesta. Automaattiset skannerit osaavat tunnistaa verkkosovelluksista löytyviä sähköpostiosoitteita ja sosiaaliturvatunnuksia, mutta tiedon arvon määrittäminen vaatii asiakkaan osallistumista [9 s.11-12].

4.2.5 CSRF

CSRF (Cross Site Request Forgery) on hyökkäys, jossa hyökkääjä pystyy hyödyntämään käyttäjän olemassa olevan istunnon oikeuksia vihamielisten komentojen suorittamiseen. Kun käyttäjä on kirjautunut tiettyyn verkkosovellukseen, lähetetään hänelle sähköpostilla linkki, jonka takana oleva ohjelma antaa käskyn jo istunnon muodostaneelle verkkosovellukselle. Tällöin näyttää, että laillisesti sovellukseen kirjautunut käyttäjä antaa komennon ja verkkosovellus ei pysty tarkistamaan komennon oikeaa alkuperää. Hyökkäys on erittäin vaarallinen, koska tällöin komento tulee käyttäjän IP-osoitteesta ja palomuuri ei pysty torjumaan hyökkäystä. CSRF-hyökkäyksiltä voidaan suojautua lisäämällä sovelluksen verkko-osoitteeseen generoitua tietoa, joka vaikeuttaa sovelluksen ulkopuolelta tulevien komentojen muodostamista. Toinen yleinen tapa on lähettää PHP-pyynnöt POST-muodossa ja käyttää GET-pyyntöä vain julkisissa palveluissa. Kolmas suojautumiskeino on toiminnon suorittamisen varmistavat ponnahdusikkunat. Tietyn toiminnon suorittamiseen lisätty vahvistuspyyntö herättää käyttäjän huomioin taustalla suoritettaville komennoille. Käyttäjä voi itse suojautua kirjautumalla heti ulos sovelluksista joita ei käytä, kytkemällä salasanojen tallennuksen pois selaimesta ja käyttämällä eri selainta verkkosurffailussa ja tärkeiden asioiden hoitamisessa [62].

4.3 Uhkien ja haavoittuvuuksien standardointi

Edellä on esitetty OWASP-järjestön luokittelu kymmenestä yleisimmästä verkossa esiintyvästä uhasta. Uhan määrittely sisältää haavoittuvuuden ja ne on listattu järjestön kotisivuilla 29:ään pääluokkaan ja 169:ään alaluokkaan. Haavoittuvuuksien määrittely on järjestön osalta kesken, joten määrät ja luokat voivat vielä muuttua. Kaikkia haavoittuvuuksia ei pystytä määrittelemään vain yhden pääluokan alle, koska yhdellä haavoittuvuudella voi olla useita eri seurauksia. Lisäksi uhat ja haavoittuvuudet kehittyvät jatkuvasti ja järjestön aikaisemmissa listauksissa osa uhkista on yhdistetty samana luokan alle ja osa on jaettu omiksi uhkaluokiksi [63].

Haavoittuvuusskannerit raportoivat löytämänsä haavoittuvuudet joko omien luokkiensa mukaan, tai käyttävät valmiita standardeja. OWASP-järjestön lisäksi haavoittuvuuksia ja uhkia on listannut CWE-projekti (Common Weakness Enumeration), joka on käyttäjien ylläpitämä listaus tietoturvahaavoittuvuuksista. Projektin yhteyteen on syntynyt CVE-listaus (Common Vulnerabilities and Exposures). CVE-listausta voidaan pitää haavoittuvuuksien sanakirjana, ja luokittelun avulla tietoa voidaan verrata eri sovellusten ja tietokantojen välillä. Listausta käyttävät tietolähteenä ainakin National Institute of Standards and Technology-virasto, joka toimii Amerikan kauppaministeriön alla, sekä US-CERT-organisaatio, joka vastaa osittain Amerikan tietoturvan valvonnasta ja kehityksestä [49]. Kolmas listaus haavoittuvuuksista on WASC-yhtymän (Web Application Security Consortium) käyttämä listaus viidestäkymmenestä verkkosovelluksia koskevasta haavoittuvuudesta. Tässä työssä käytettävistä haavoittuvuusskannereista toinen ilmoittaa haavoittuvuudet oman standardinsa mukaan ja toinen käytti CVE- ja WASC-koodeja.

5 VERKKOSOVELLUSTEN TIETOTURVATESTAUS

Tässä luvussa käsitellään verkkosovellusten testaukseen tarvittava järjestelmä, käytetyt ohjelmat ja pohditaan testauksen suorittamiseen liittyviä asioita. Diplomityötä varten testaukseen valittiin kuusi kappaletta dHosting-verkkohotellipalvelussa toimivaa verkkosovellusta. Valintaa varten jokainen sovelluksen määritelmän täyttävä verkkosivu listattiin ja niiden omistajilta pyydettiin lupa suorittaa testit sovellukselle suljetussa ympäristössä. Jotta testausohjelmien toiminta voidaan varmistaa ja saadaan vertailtavia tuloksia, asennettiin verkkopalvelimelle lisäksi RandomStorm-nimisen yrityksen julkaisema Damn Vulnerable Web Application (DVWA) [74]. Sovelluksen tarkoitus on simuloida huonosti ohjelmoitua verkkosovellusta ja sitä voidaan käyttää haavoittuvuuksien simulointiin sekä testausohjelmien harjoitteluun. Tässä kappaleessa esitettävät esimerkit on suoritettu DVWA-sovellukselle. Jos sovellus tarjoaa mahdollisuuden kirjautua, luodaan testausprosessia varten käyttäjätason tunnukset ja testaus suoritetaan käyttäjän näkökannalta. DVWA-sovelluksella esitetyt testit suoritetaan kaikille testaukseen valituille sovelluksille ja tulokset on esitetty luvussa 6. Testauksena mukana olleiden sovellusten verkko-osoitteita ei mainita asiakkaiden tietoturvan takaamiseksi.

Testausta suunniteltaessa päädyttiin testausjärjestelmään, jossa testattavat verkkosovellukset kopioidaan paikalliselle koneelle, johon asennetaan VirtualBox-virtualisointiohjelma. Ratkaisuun päädyttiin, koska tuotantopalvelimella olevien sovellusten testaus saattaisi vaarantaa kohdesovelluksen saavutettavuuden. VirtualBox-ohjelmalla luodaan suljettu virtuaalinen lähiverkko, joka ei ole yhteydessä internetiin. Lähiverkossa toimii kaksi virtuaalista konetta, joista toinen toimii verkkopalvelimena ja toinen hyökkääjänä. Verkkopalvelimelle asennetaan kopiot testaukseen valituista verkkosovelluksista. Hyökkääjää esittävälle virtuaalikoneelle asennetaan Linux Kali-käyttöjärjestelmä ja tältä koneelta suoritetaan halutut testit ja hyökkäyksen. Tuotantokäytössä olevasta verkkohotellista on mahdollista ottaa kopio ja asentaa toiselle virtuaalipalvelimelle, mutta tähän tarvittavien resurssien varaaminen ei ollut mahdollista diplomityön puitteissa. Lisäksi asiakastietojen kopioiminen ilman kaikkien palvelussa olevien asiakkaiden suostumusta voisi aiheuttaa sopimusteknisiä ongelmia. Testaus päädyttiin suorittamaan sovelluskohtaisesti ja asiasta sovittiin jokaisen asiakkaan kanssa erikseen. Huomioitavaa on, että sovellusten skannausta ja hyökkäyksiä ei suorita testauksen ammattilainen, vaan tietotekniikan kohtalaisesti tunteva käyttäjä. Tässä testauksessa ei löydetä kaikkia sovellusten haavoittuvuuksia ja tarkoitus on saada näkymä testaukseen ja haavoittuvuuksiin yleisellä tasolla.

Luvussa käsitellään myös testaukseen valitut ohjelmistot. Kali on Linux-käyttöjärjestelmän julkaisuversio, joka sisältää tietoturvatestaukseen liittyviä ohjelmia ja työkaluja. Verkkosovelluksen tunnistamisessa päädyttiin käyttämään BlindElephant [52] ja Plecost [69] nimisiä ohjelmia, sekä verkkopohjaisia työkalua. Haavoittuvuuksien skannaukseen käytetään Vega [72] ja OWASP ZAP [53] nimisiä ohjelmia. Suuri osa verkkosovelluksiin kohdistuneista hyökkäyksistä on automatisoituja tietokoneiden suorittamia skannauksia, jotka etsivät entuudestaan tunnettuja haavoittuvuuksia. Testauksen on tarkoitus mallintaa tällaista tilannetta ja testejä ei räätälöidä sovelluskohtaisesti. Jos työssä löydetään testaukseen sopiva ja automatisoitavissa oleva työkalu, voidaan se ottaa mahdollisesti osaksi verkkohotellipalvelun lisäpalveluja. Tavoite on tarjota asiakkaalle mahdollisuus selvittää, toteuttaako heidän sovelluksensa hyvää suunnittelua, herättää keskustelua tietoturvasta ja mahdollisesti ennakoida ja vähennetään palveluntarjoajalle tulevien tukipyyntöjen määrää.

5.1 Testauksen aloittaminen ja siihen liittyvät sopimukset

Tietoturvatestaus on suositeltavaa aloittaa sopimalla testauksen rajat asiakkaan kanssa. Ongelmatilanteita varten sopimus tehdään mielellään kirjallisesti. PTES (Penetration Testing Exection Standard) on järjestö jonka tarkoitus on luoda ohjeet testausprosessin suunnitteluun [65]. Ohjeiden mukaan sopimukseen sisällytetään tiedot milloin testaus aloitetaan, milloin lopetetaan ja suoritetaanko se tiettynä ajanjaksona vuorokaudesta tai viikosta. Lisäksi sopimuksessa rajataan minkä IP-osoitteen tai verkko-osoitteen takana kohteet sijaitsevat. Testattavassa kohteessa on mahdollisesti upotettu kolmannen osapuolen sovelluksia ja palveluita. Pilvipohjaisia ja omia palveluita yhdistelevissä sovelluksissa asiakas ei ole aina tietoinen, tai muista, mitä komponentteja heidän sovelluksensa sisältää. Myös internetyhteyden palveluntarjoaja, verkkohotellipalveluntarjoaja, pilvipalveluntarjoaja, sekä kansainväliset lait täytyy huomioida testausta suunniteltaessa. Jos testaus sisältää häiriötä aiheuttavia rasitukseen perustuvia testejä, sovitaan oikeudet ja vastuut ennalta. Testauksen suunnitteluun sisältyy lisäksi maalien asettaminen. Muuttuvien ympäristöjen testaus vaatisi jatkuvaa testausta, ja kertaluontoisella testauksella saadaan tietoturvan tasoon vain hetkellinen näkymä. Testauksen valmistuminen voidaan määritellä löytyneiden haavoittuvuuksien määränä, käytettyjen tekniikoiden ja sovellusten listauksella, tai perustuen tietyn pituiseen ajanjaksoon. Tässä työssä käytettyjen sovellusten listaus toimii testauksen maalina. Maalit sovitaan testauksen määrittelydokumentissa ja tällöin asiakkaalle voidaan lopussa osoittaa testauksen valmistuminen. Sovellustestausta voidaan osittain siirtää myös asiakkaiden tehtäväksi. Osa verkossa toimivista yrityksistä maksaa käyttäjilleen haavoittuvuuksien löytämisestä ja näin parantaa omaa tietoturvaansa.

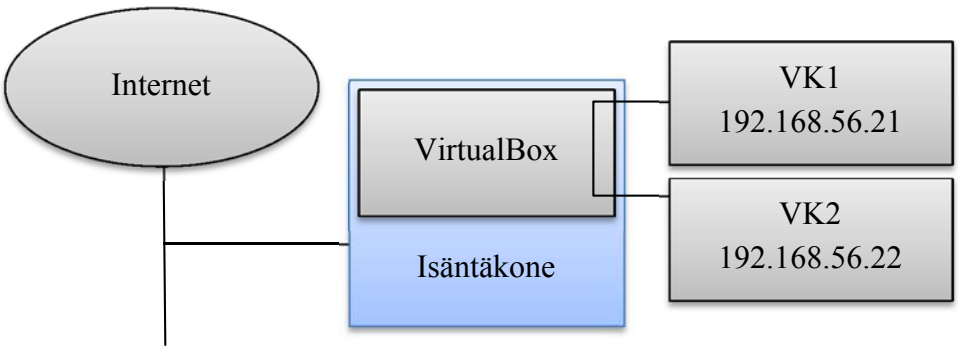
5.2 Työkalut

Tässä kappaleessa esitellään testaukseen käytettyjä ohjelmia ja käydään läpi niiden toiminta DVWA-sovelluksella. Skannausohjelmien valinta perustuu OWASP-järjestön listaukseen ei-kaupallisista haavoittuvuusskannereista, joista valitut kaksi ovat valmiiksi asennettu Kali-käyttöjärjestelmään [66].

5.2.1 VirtualBox

VirtualBox on Windows-, Mac-, Solaris, tai Linux-käyttöjärjestelmässä toimiva virtualisointiohjelma. Sen avulla ensisijaisen käyttöjärjestelmän päälle voidaan luoda virtuaalisia tietokoneita, eli virtuaalikoneita. Virtuaalikoneet jakavat verkko-, IO- ja laskentaresurssit isäntäkoneen kanssa. Ohjelmalla voidaan luoda erilaisia verkkotopologioita, joista kolme yleisintä ovat osoitteenmuutos, siltaus ja suljettu verkko. Tässä työssä verkko toteutetaan suljettuna verkkona, jossa on kaksi virtuaalikonetta. Tällöin vihamielisiä ohjelmia voidaan ajaa suljetussa ympäristössä ja ne eivät pääse vaikuttamaan ulkopuolisiin laitteisiin tai verkkoon. Tarvittaessa virtuaalikone voidaan palauttaa tiettyyn ennalta tallennettuun tilaan jos testaus tai hyökkäys vahingoittaa sen pois toimintakunnosta [67].

Testausympäristön rakenne on kuvattu kuvassa 5.1.



Kuva 5.1. VirtualBox sovellukseen testausta varten luodut verkkoasetukset

Virtuaalikone-1 toimii verkkopalvelimena verkko-osoitteessa 192.168.56.21 ja siihen on asennettu käyttöjärjestelmä, Apache-verkopalvelin, PHP-tulkki ja MySQL-tietokantasovellus. Lisenssisyistä koneeseen ei asenneta verkkohotellin hallintapaneelia. dHosting-palvelussa sijaitsevista asiakassovelluksista tehdään kopio ja siirretään toimimaan suljettuun ympäristöön. Virtuaalikone-2 on Linux Kali-asennus, joka pystyy virtuaalisen lähiverkon kautta ottamaan yhteyden Virtuaalikone-1:llä toimivaan verkkopalvelimeen. Suljettuun verkkoon päädyttiin, koska asiakassovelluksia, eikä testauksessa käytettäviä ohjelmia, tunneta tarpeeksi hyvin. Jos asiakassovellus olisi yhteydessä verkkoon, voisi se lähettää sähköpostia ylläpitäjälleen hyökkäysyrityksistä, tai samoin hyökkäyssovellus voisi kerätä ja lähettää tietoja ohjelmoijilleen kohdesovelluksesta. Verkkoyhteys mahdollistaisi lisäksi ulkoa tulevat hyökkäykset verkkopalvelinta kohtaan. Tällä järjestelyllä voi-

daan varmistua, että asiakkaiden alkuperäisien sovelluksien saatavuus tai tiedon luottamuksellisuus ei vaarannu missään vaiheessa. Riskinä järjestelmässä on asiakassovelluksien sijaitseminen kahdessa kohteessa, jolloin testauksessa käytettävän isäntäkoneen kaappaus mahdollistaisi tietojen vuotamisen.

5.2.2 Linux Kali

Linux Kali on tietoturvatestaukseen erikoistunut ilmainen Debian-pohjainen Linux-versio, joka sisältää yli 300 testaukseen tarkoitettua ohjelmaa tai työkalua. Ensimmäinen versio käyttöjärjestelmästä julkaistiin vuonna 2004 nimellä BackTrack. Uusin versio käyttöjärjestelmästä kirjoitushetkellä on julkaistu 9. tammikuuta 2014 julkaistu Linux Kali. Ilmaistyökalut ovat tehokkaita, mutta syvällisempää testaukseen suositellaan juuri tehtävään tarkoitettuja, mahdollisesti maksullisia, työkaluja. Kalin on julkaissut Offensive Security-niminen yritys [68]. Yrityksen liiketoiminta perustuu ohjelman ilmaiseen jakeluun, mutta tarjoaa sen käyttöön hintavia koulutuksia. Lisenssiehdoissa määritetään, että kotikäyttöön ja yrityksien sisäinen testaus on sallittua, mutta kaupalliseen käyttöön lisensointi täytyy sopiva Offensive Security:n kanssa.

Kalilla on laaja käyttäjäkunta ja sen käyttöön löytyy verkosta kattavasti opetusmateriaalia. Käyttöjärjestelmää on ladattu miljoonia kertoja ja sen on tällä hetkellä käyttäjämäärällä mitattuna markkinoiden suosituin tietoturvan testausympäristö. Linux Kalia käytetään lailliseen testaukseen, mutta sen avulla voidaan suorittaa myös vihamielisiä hyökkäyksiä. Kali voidaan asentaa koneen käyttöjärjestelmäksi, ajaa muistikortilta tai CD:ltä, tai asentaa esimerkiksi VirtualBoxin-päälle. Kali on saatavissa osoitteesta www.kali.org.

5.2.3 Sovelluksen tunnistamiseen käytettävät ohjelmat

Koska useat verkkosovellukset rakennetaan julkaisujärjestelmien avulla, voidaan ne tunnistaa ennalta tunnettujen tiedostojen tai rakenteiden perusteella. Usein ensimmäinen vaihe hyökkäyksestä on tiedon kerääminen kohteesta. Kun kohdejärjestelmä on tunnistettu, voidaan hyökkäys kohdentaa sen tunnettuihin haavoittuvuuksiin. Kali-käyttöjärjestelmässä on kolme tähän tarkoitukseen tarkoitettua ohjelmaa, joista tässä työssä käytetään kahta.

BlindElephant on komentorivipohjainen työkalu ja sen tunnistaa kohdesovelluksen viidestätoista ennalta määritetystä sovelluksesta. Näihin kuuluvat yleisimmät julkaisujärjestelmät, wiki-ohjelmat ja blogi-alustat. Ohjelma voidaan käynnistää komennolla:

```
BlindElephant.py 192.168.56.21/dvwa/ guess
```

Kalin komentorivillä annetaan komento ajaa BlindElephant.py-niminen python-kielinen ohjelma. Ellei kohdesovelluksesta ole tarkempaa tietoa, annetaan ohjelmalle parametreiksi analysoitavan verkkosovelluksen verkko-osoite ja guess-komento. Vastaukseksi ohjelma tulostaa arvion kohteen toteutukseen käytetystä julkaisujärjestelmästä. Koska

DVWA-sovellus ei ole toteutettu millään ohjelman tuntemista ympäristöistä, ei sitä tunnisteta.

Kaksi muuta sovellusta liittyvät WordPress-julkaisujärjestelmän versioiden ja sen lisämoduulien tunnistamiseen. Käyttökokeilun perusteella testaukseen valittiin Plecost-niminen ohjelma, joka tunnistaa eri WordPress-versiot ja käytössä olevat lisäosat tiedostorakenteen, linkkien ja Googlen indeksoinnin perusteella [69]. Kohdesovelluksen skannaus aloitetaan komennolla:

```
/usr/bin/plecost –i –h 192.168.56.21/dvwa/
```

Samoin kuin BlindElephant, Plecost-sovellus palauttaa komentorivitulosteen tunnistamastaan WordPress-versiosta ja siinä käytössä olevista lisämoduuleista. Koska DVWA ei ole WordPress-pohjainen sovellus, ei sitä tunnisteta tällä ohjelmalla. Molempien ohjelmien tulokset muille testattaville sovelluksille ovat esitetty kappaleessa 6.1. Kun kohdesovellus on mahdollisesti tunnistettu ja kohteesta on luotu yleiskuva, voidaan siirtyä haavoittuvuuksien tunnistamiseen.

5.3 Haavoittuvuusskannerit

Verkkosovellusten haavoittuvuusskannerit ovat automatisoituja työkaluja, jotka tutkivat verkkosovelluksia ja etsivät niistä haavoittuvuuksia. Haavoittuvuusskannerit kuuluvat BlackBox-testauksen työkaluihin, koska niillä ei ole pääsyä sovelluksen lähdekoodiin. Sovellukset kartoitetaan löytyneiden linkkien perusteella ja haavoittuvuuksien havaitseminen tapahtuu joko muokkaamalla HTTP-viestejä tai syöttämällä sopivasti muokattuja syötteitä. Haavoittuvuusskannereita on maksullisia ja avoimeen lähdekoodiin perustuvia ja molemmat tässä työssä käytetyistä sovelluksia ovat ilmaisia Kaliin integroituja sovelluksia. Haavoittuvuusskannerit eivät löydä kaikkia sovelluksen virheitä, mutta niiden avulla voidaan luoda nopeasti kuva sovelluksen yleisestä tietoturvasta. Löytyneet haavoittuvuudet on suositeltavaa testata tarkemmin sovelluskehittäjän toimesta. Osa haavoittuvuuksista on todennäköisesti virheellisiä positiivisia, jolloin haavoittuvuusskanneri antaa virheellisen hälytyksen haavoittuvuutta muistuttavasta ominaisuudesta.

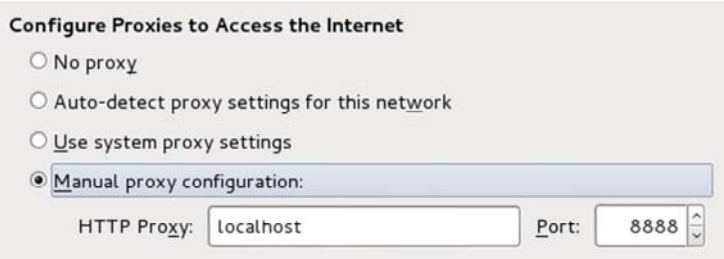
WASSEC (Web Application Security Scanner Evaluation Criteria) on dokumentti, joka luokittelee sovellusskannereita kuuden ominaisuuden perusteella. Hyvällä verkkoskannerilla täytyy olla tuki usealle siirtoprotokollalle, mahdollisuus kirjautumiseen ja istunnonhallintaan, ominaisuus verkkosovelluksen kartoittamiseen, kääntämiseen ja testaukseen, sekä automatisoitu raportointi. Fakhreldeen Saeed on vertaillut eri avoimen lähdekoodin haavoittuvuusskannereita IJCSN-lehden artikkelissa [70] ja hän suosittelee käytettäväksi IronWASP-, W3AF-, SkipFish-, arachniv- ja OWASP ZAP-sovelluksia. Näistä OWASP ZAP-sovellus on valmiina asennettu Kali-käyttöjärjestelmään ja valitaan toiseksi testausohjelmaksi. Toiseksi ohjelmaksi valitaan Vega-niminen haavoittuvuusskanneri. Sovelluksen saamat pisteet vertailussa olivat hyvin lähellä suositeltuja

viittä sovellusta, se on valmiiksi asennettu Kaliin, ja siinä on helppo ja selkeä käyttöliittymä. Lisäksi internetistä löytyi kattavasti ohjeita ohjelman käyttöön.

5.3.1 Vega

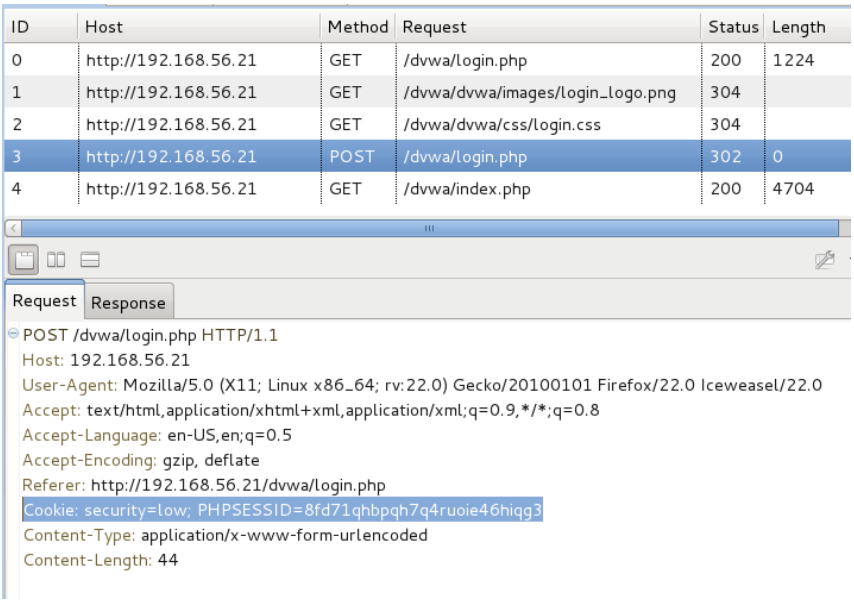
Vega on Kali-käyttöjärjestelmään integroitu avoimeen lähdekoodiin perustuva verkkosovellusten haavoittuvuusskanneri. Ohjelma tunnistaa SQL-injektioita, XSS-haavoittuvuuksia, huonosti suunniteltuja ohjelmakoodeja, mahdollisesti arkaluontoista materiaalia, sekä muita haavoittuvuuksia. Vega on kirjoitettu Java-ohjelmointikielellä ja siinä on graafinen käyttöliittymä. Vega toimii testauksen aikana välittäjäpalvelimena ja se nauhoittaa kaikki lähetyt ja vastaanotetut HTTP-viestit muistiin myöhempää analyysia varten [71].

Ohjelman käyttö aloitetaan määrittelemällä selainohjelman välittäjäpalvelin selaimen verkkoasetuksista. Kuvassa 5.2. on esitetty Kalissa oletuksena käytettävän Iceweasel-selaimen verkkoasetukset.



Kuva 5.2. Selainohjelman välittäjäpalvelinasetukset Vega-haavoittuvuusskannerille

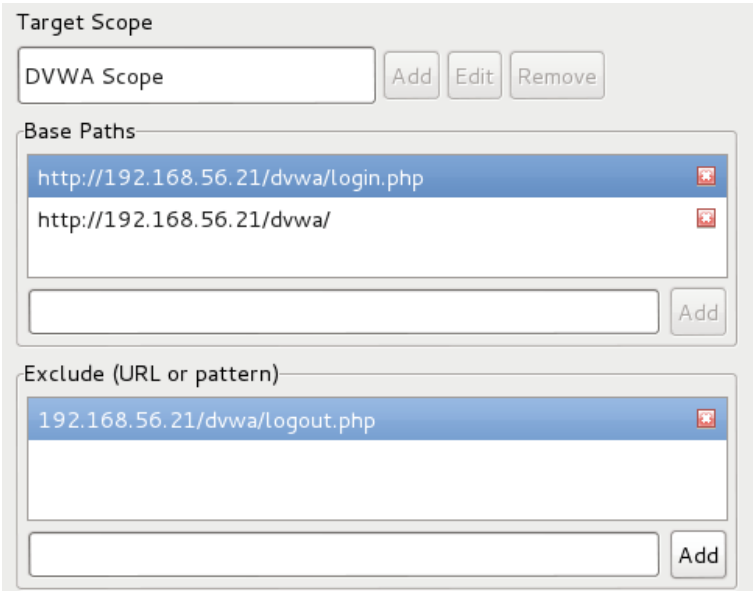
Asetuksissa selain määritetään suorittamaan kaiken verkkoliikenteen Vegassa toimivan paikallisen välittäjäpalvelimen verkkoportin 8888 kautta. Tällöin kaikki selaimelta lähtevät pyynnöt ja palvelimelta tulevat vastauksen kulkevat Vegan kautta. Välittäjäpalvelimen nauhoittamat viestit on esitetty kuvassa 5.3.



Kuva 5.3. Vegan välittäjäpalvelimen tallentamat HTTP-viestit

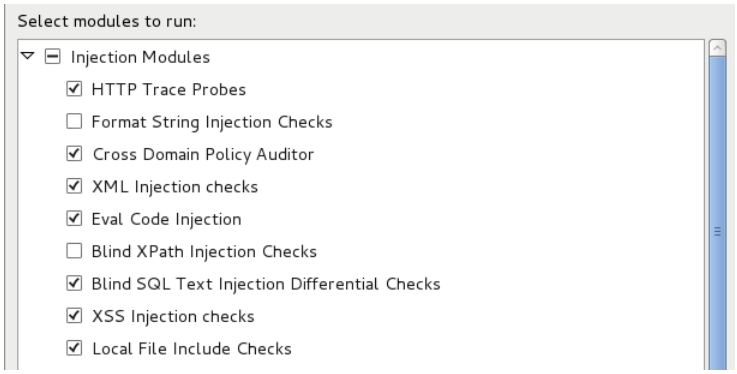
Kun selaimella siirrytään lähiverkossa sijaitsevan verkkopalvelimen osoitteeseen 192.168.56.21/dvwa/, lataa selain sivuilta login.php-tiedoston, sivun logon png-kuvatiedostona ja CSS-tyylitiedoston. Login.php-sivulta verkkosovellukselle lähetetään POST-komennolla käyttäjätunnus ja salasana. Verkkosovellus käsittelee vastauksen ja palauttaa viestin jossa on mukana istuntotunniste. Tämä viesti luetaan Vagassa olevalla työkalulla kirjautumiseen käytettäväksi makroksi, jota käytetään skannausvaiheessa tunnistautumiseen.

Skannauksen aloittamisen yhteydessä Vegalle määritetään skannattava kohde. Tämä tehdään asettamalla testattavan verkkosovelluksen näkyvyysala. Kuvassa 5.4 on esitetty DVWA-sovelluksen skannaamiseen käytetyt asetukset.



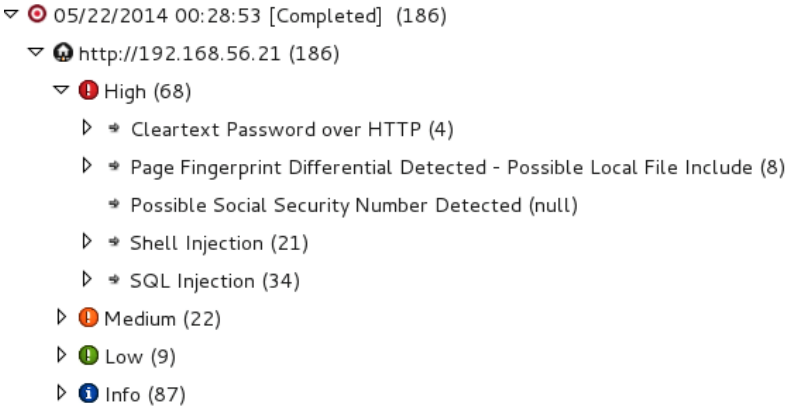
Kuva 5.4. Haavoittuvuusskannerin näkyvyysalueen määrittely

Kuvasta huomataan, että näkyvyysala sisältää kohdesovelluksen kotisivun ja sisäänkirjautumiseen tarkoitetun verkkosivun. Sovelluksen analysointi aloitetaan annetuista osoitteista ja sivuilla olevia linkkejä seuraamalla kartoitetaan verkkosovelluksen rakenteen. Osa kohdeverkkosovelluksen linkeistä osoittaa 192.168.56.21/dvwa/-verkkoalueen ulkopuolelle ja ne jätetään huomioimatta. Sovelluksesta ulos kirjautuva linkki tai tiedosto on suositeltavaa jättää näkyvyysalueen ulkopuolelle, ettei skanneri kirjaa itseään vahingossa ulos kohdesovelluksesta. Seuraava vaihe skannauksen alustamisessa on määrittä testauksessa käytettävät moduulit. Eri moduuleja Vegassa on noin kaksikymmentä ja osa valikoimasta on esitetty kuvassa 5.5.



Kuva 5.5. Haavoittuvuusskannerin moduulien valinta

Halutut moduulit valitaan listasta ja määrä vaikuttaa skannauksen kestoon. Testeissä noin 2000-sivua ja tiedostoa sisältävän sovelluksen skannaus kesti useita tunteja. Kun Vega on käynyt läpi verkkosovelluksen rakenteen ja analysoinut siellä sijaitsevat mahdolliset haavoittuvuudet, saadaan tästä haavoittuvuudet vaarallisuuden ja tyypin mukaan valmiina raporttina. Kuvassa 5.6 on esitetty Vegan antama malliraportti DVWA-sovellukselle.



Kuva 5.6. Löydettyjen haavoittuvuuksien raportointi Vega-haavoittuvuusskannerissa

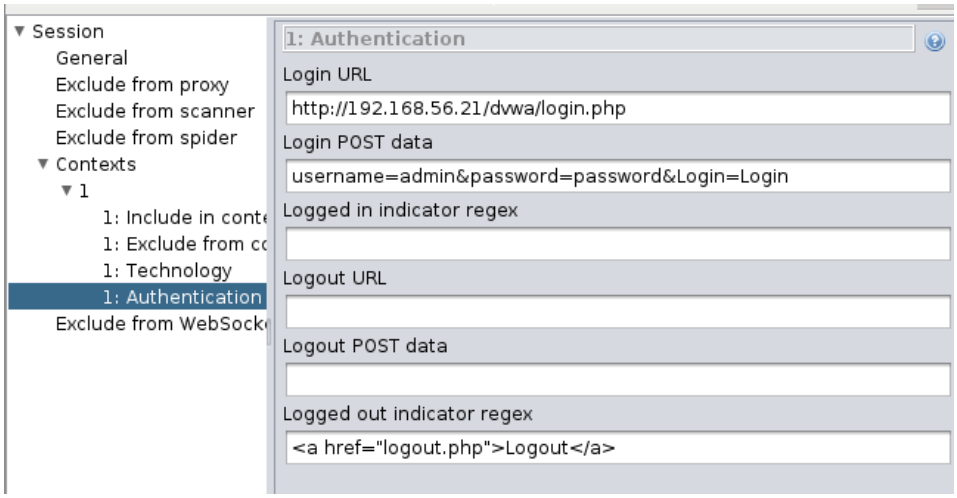
Haavoittuvuudet jaetaan kriittisen-, keski-, matalan- ja infoluokan haavoittuvuuksiin. Jokainen raportoitu haavoittuvuus on avattavissa omaan välilehteen, jossa tarjotaan tietoa mistä tiedostosta haavoittuvuus löydettiin, kuvaus itse haavoittuvuudesta, miten se voi vaikuttaa sovellukseen, miten sovelluskehittäjien tulisi suojautua haavoittuvuudelta ja lähdemateriaalia tarkempaa analyysia varten. Näitä tietoja käytetään luvussa 6 avuksi eri haavoittuvuuksien analysointiin [72].

5.3.2 OWASP ZAP

OWASP Zed Attack Proxy (OWASP ZAP) on OWASP-järjestön kehittämä verkkosovellusten haavoittuvuusskanneri. Ohjelma on valittu toolswatch-org-sivuston käyttäjääänestyksessä vuoden 2013 parhaaksi tietoturvasovellukseksi. Ohjelma on ilmainen ja perustuu avoimeen lähdekoodiin. Se toimii Windows, Linux ja OS X-käyttöjärjestelmissä ja on integroitu Kali-käyttöjärjestelmään [73].

Sovellus sisältää passiivisia ja aktiivisia skannereita. Passiivinen skanneri analysoi lähetettyjä ja vastaanotettuja viestejä ja tunnistaa niistä mahdollisesti vaarallisia malleja. Passiivista skanneria on mahdollista käyttää myös verkossa toimivien sivujen analysointiin ilman sivuston ylläpitäjän lupaa. Esimerkki tällaisista skannereista ovat spider-sovellukset, jotka tutkivat ja kartoittavat verkkosivujen rakennetta. Skannerin toiminta on verrattavissa käyttäjään, joka etsii ja kirjaa ylös kaikki sivuilta löytyvät linkit. Aktiivinen skanneri sen sijaan suorittaa erilaisia testejä ja hyökkäyksiä kohdesovellukselle ja testausprosessi täytyy aina sopia verkkosovelluksen ylläpitäjän kanssa. OWASP-sovellus sisältää lisäksi Brute Force-tekniikkaa käyttäviä komponentteja, jotka etsivät ei linkitettyjä piilosivuja, sekä Fuzzing-tekniikkaan perustuvia komponentteja, jotka lähettävät verkkosovelluksille satunnaista tai jonkin algoritmin perusteelle luotua häiriötä ja virheitä aiheuttavia syötteitä.

Ohjelman käyttö aloitetaan asettamalla sovellus selainohjelman välittäjäpalvelimeksi verkkoporttiin 8080. Ohjelmassa on automaattinen toiminto, joka suorittaa tämän käyttäjän puolesta yleisimmille selaimille. Seuraava vaihe on kohdesovelluksen tunnistautumistietojen määrittely. Testaaja kirjautuu verkkosovellukseen ja välittäjäpalvelin nauhoittaa verkkosovelluksesta tunnistautumiseen tarvittavan HTTP-viestin. Nauhoitetun viestin pohjalta luodaan testaukseen tarvittava istuntotunniste, jota käytetään skannauksen aikana.



Kuva 5.7. OWASP ZAP-haavoittuvuusskannerin kohdesovelluksen kirjautumistietojen määrittely

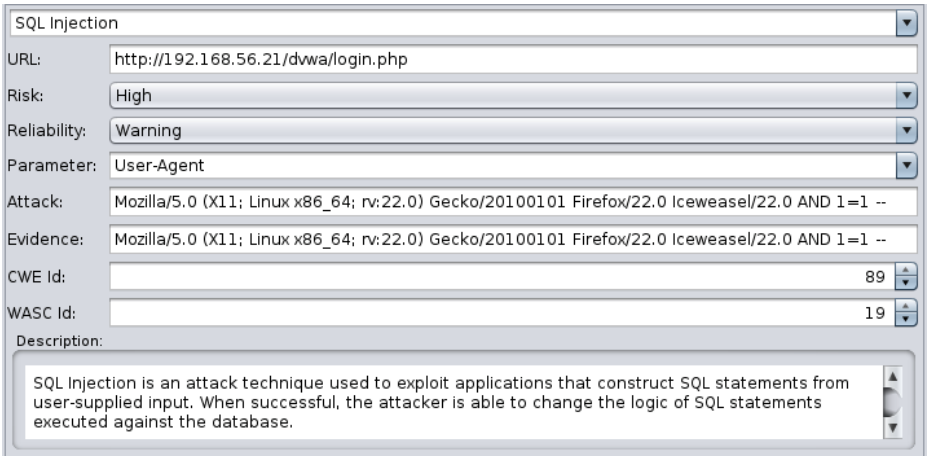
Seuraavassa vaiheessa haavoittuvuusskannerista valitaan Spider-välilehti ja ohjelmalle annetaan komento kartoittaa kohdesovellus. Kun kohdesovelluksen tiedostorakenne on kartoitettu, aloitetaan haavoittuvuuksien etsiminen. Testausprosessin aikana ohjelman asetuksiin tutustuttiin, mutta niitä ei koettu tarpeelliseksi muokata ja skannausvaihe suoritettiin oletusasetuksilla. OWASP ZAP suorittaan kohdesovelluksen skannauksen selvästi nopeampi kuin Vega. Yhden sovelluksen kartoittaminen ja skannaaminen kestää noin 5 minuuttia, kun Vegassa vastaava toimenpide oli puolesta tunnista kahteen tuntiin.

Skannerin nopeutta voidaan muuttaa sovelluksen asetuksista. Tämä on hyödyllinen ominaisuus tilanteissa, jossa skannauksella ei haluta herättää sovelluksen ylläpidon tai palomuurin huomiota.

Skannauksen jälkeen ohjelma luo raportin löytyneistä haavoittuvuuksista. Sovelluksen raportti löytyneistä haavoittuvuuksista on esitetty kuvassa 5.8.

Kuva 5.8. **OWASP ZAP-haavoittuvuusskannerin raportti löytyneistä haavoittuvuuksista**

Kuvasta huomataan, että OWASP ZAP kuvaa eri tason haavoittuvuudet erivärisinä lippuina, mutta käyttää samaa luokittelua vakavuuden tasosta kuin Vega. Jokainen haavoittuvuus voidaan avata omaksi raportiksi ja tällainen raportti on kuvattu kuvassa 5.9.



Kuva 5.9. **SQL-injektioin haavoittuvuus raportti**

Ohjelman antaa tiedon mihin dokumenttiin tai tiedostoon haavoittuvuus viittaa, sen riskiluokituksen ja lisätietoja haavoittuvuudesta. OWASP ZAP-ohjelma ilmoittaa lisäksi mihin CWE- tai WASC-luokan haavoittuvuuteen tai uhkaan haavoittuvuus perustuu.

5.4 Verkkopohjaiset työkalut

Aikaisemmin esitellyt työkalut toimivat paikallisessa testausympäristöön ja niitä käytetään verkkosovelluksista tehtyjen kopioiden analysointiin. Internetistä löytyy useita verkkosovelluksien analysointiin tarkoitettuja sovelluksia, jotka toimivat samoin kuin haavoittuvuusskannerien kartoitusvaihe. Näillä sovelluksilla saada selville mitä julkaisujärjestelmää sovelluksen taustalla käytetään, mitä viitekehyksiä sovelluksella on käytössä ja muuta kohteeseen liittyvää informaatiota. Verkkopohjaisia skannereita voidaan käyttää asiakkaiden sovellusten tutkimiseen ilman asiakkaan lupaa, koska ne mallintavat tilannetta, joka on vastaava, kuin käyttäjän vierailu verkkosivuilla. Tähän työhön on valittu kaksi tällaista sovellusta jotka ovat Sucuri (sitecheck.sucuri.net) ja Guess (http://guess.scritch.org/). Kumpikin sovellus toimii kohdesovelluksen URL-osoitteen perusteella. Verkko-osoite annetaan syötteenä verkkosivuilla, ja tämän jälkeen sovellus tulostaa raportin löytämistään tiedoista.

Toinen tapa kartoittaa verkossa toimiva sivustoja on selaimeen asennettavat lisäosat. Tässä työssä FireFox-selaimeen asennettiin Wappalyzer-lisäosa, joka analysoi jokaisen selaimella avatun verkkosivun ja sovelluksen. Selaimen osoitekenttään ilmestyvistä kuvakkeista saadaan lisätietoa vierailtavan sivun rakenteesta. Saatavat tiedot ovat vastaavia, kuin sivuja analysoivissa sovelluksissa. Verkkopohjaisia työkaluja käytetään kuuden kohdesovelluksen tunnistamiseen tuotantoympäristössä, sekä niillä analysoidaan kaikki dHosting-palvelussa olevat verkkosivut. Näin saadaan tarkempi kokonaiskuva palvelussa olevista sivuista ja asiakaskannan rakenteesta. Tätä tietoa käyttämällä voidaan arvioida miten valitut kuusi kohdesovellusta kuvastavat koko palvelun sisältöä.

6 TULOKSET

Tässä luvussa esitetään kappaleessa viisi esiteltyjen ohjelmien ja sovellusten avulla saadut testaustulokset. Testit suoritettiin kuudelle eri asiakassovellukselle. Mukana testiryhmässä oli lisäksi DVWA-verkkosovellus, joka toimii testeissä vertailukohteena. Testattavien kohteiden määrä jäi matalaksi, koska verkkohotellipalvelu on vasta perustettu ja asiakaskanta on tästä johtuen suppea. Useassa tapauksessa asiakas ei antanut suostumusta testaukseen, koska verkkosovellukset sisälsivät heidän asiakkaidensa tietoja, joita ei haluttu antaa osaksi kolmannen osapuolen testausta. Jokainen kontaktoitu asiakas piti testaukseen liittyvää asiaa tärkeänä ja näki tietoturvaan liittyville lisäpalveluille mahdollisesti käyttöä.

6.1 Tulosten esittely

Tässä kappaleessa esitetään testausohjelmilla ja -sovelluksilla saadut testitulokset. Kohdesovelluksien nimiä ei mainita tietoturva syistä, vaan niintä käytetään järjestysnumeroa.

6.1.1 Sovelluksen tunnistaminen

Sovelluksen tunnistamiseen käytettiin BlindElephant- ja Plecost-ohjelmia, sekä kolmea internetissä toimivaa sovellusta. Tulokset on esitetty taulukossa 6.1. Taulukossa on ilmoitettu lisäksi oikea sovelluksen taustalla toimiva järjestelmä.

Taulukko 6.1 Esimerkki valtuuttamiseen liittyvistä rooleista

Analysaattori	Sovellus 1	Sovellus 2	Sovellus 3	Sovellus 4	Sovellus 5	Sovellus 6
Oikea	WP 3.8.1	WP 3.9.1	CakePHP	WP 3.9.1	MODX	WP 3.8.3
BlindElephant	WP 3.4.3	Ei tunnistusta	Ei tunnistusta	Ei tunnistusta	Ei tunnistusta	WP 3.4.3
Plecost	WP 3.8.1	WP 3.9.1	Ei tunnistusta	WP 3.9.1	Ei tunnistusta	WP 3.8.3
Sucuri Sitecheck	WP 3.8.1	WP 3.9.1	CakePHP	WP 3.9.1	Ei tunnistusta	WP
Guess	WP	WP 3.9.1	CakePHP	WP 3.9.1	Ei tunnistusta	WP
Wapplyzer	WP	WP 3.9.1	CakePHP	WP 3.9.1	Ei tunnistusta	WP 3.8.2

Taulukosta huomataan, että neljä sovellusta kuudesta on toteutettu Wordpress-julkaisujärjestelmällä. CakePHP ei ole julkaisujärjestelmiä, vaan PHP-viitekehys, joka avustaa PHP-sovellusten ohjelmointia. Testiympäristö on painottunut Wordpress-sovelluksiin, mutta on linjassa taulukon 2.1 tulosten kanssa, joiden mukaan 60 prosenttia julkaisujärjestelmistä on Wordpress-toteutuksia. Sovelluksia valittaessa julkaisujärjestelmä tai mahdollinen viitekehys ei ollut tiedossa.

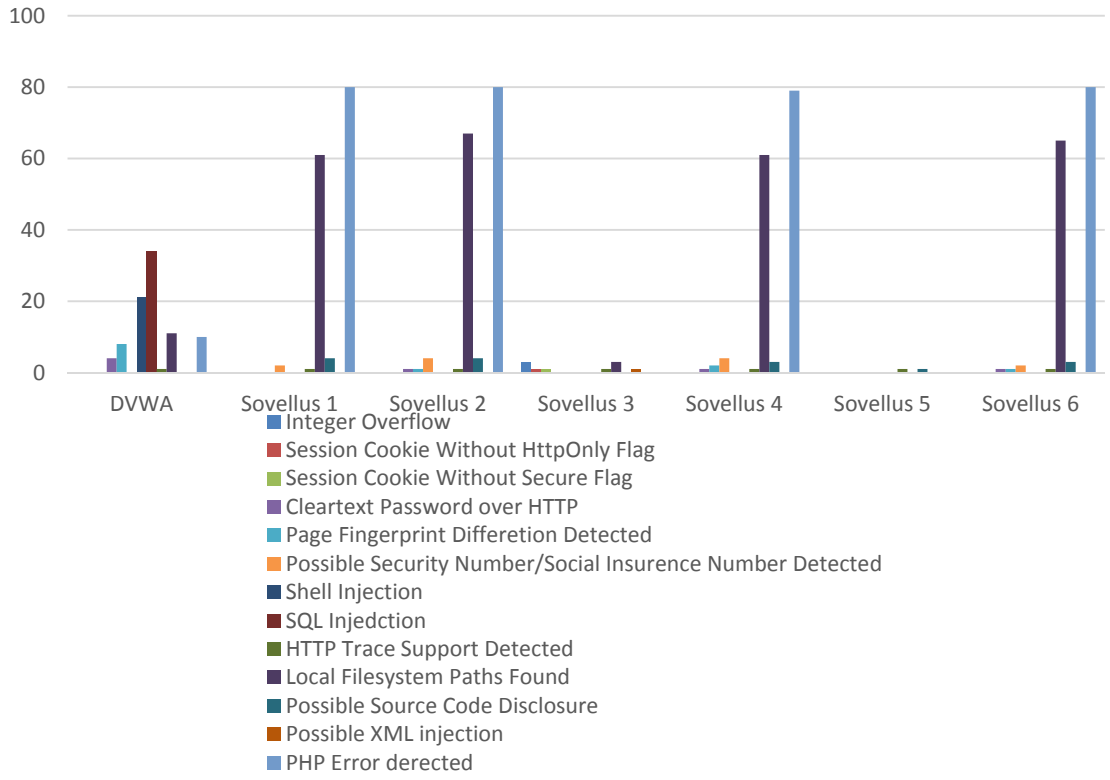
Tuloksista nähdään, että BlindElephant-ohjelma ei tunnista yhtään sovellusta oikein ja Plecost-ohjelma antaa tarkimman tunnistuksen Wordpress-versiosta. Verkossa

toimivien testaussovelluksien tulokset ovat melkein identtisiä ja näin pienellä testiryhmällä ei saada aikaan huomattavaa eroa niiden kesken. Verkkopohjaiset sovellukset tunnistavat kohdesovelluksia monipuolisimmin kuin asennetut ohjelmat, mutta vanhempien Wordpress-versioiden tunnistuksessa Plecost-sovellus antaa parhaan tuloksen. Usein hyökkäys perustuu juuri tietyn kohdesovellusversion määrätyssä ympäristössä esiintyvään haavoittuvuuteen, joten tarkka tunnistus voi olla vaatimus hyökkäyksen onnistumiselle. Näiden tulosten pohjalta on suositeltavaa käyttää verkkopohjaisia tunnistussovelluksia, koska ne ovat nopeampia käyttää, tunnistavat sovelluksia laajemmin ja ne eivät vaadi asentamista. Jos halutaan vertailla eri tunnistusohjelmia, niin testaus laajemmalla lähdemateriaalilla olisi järkevää.

6.1.2 Vega

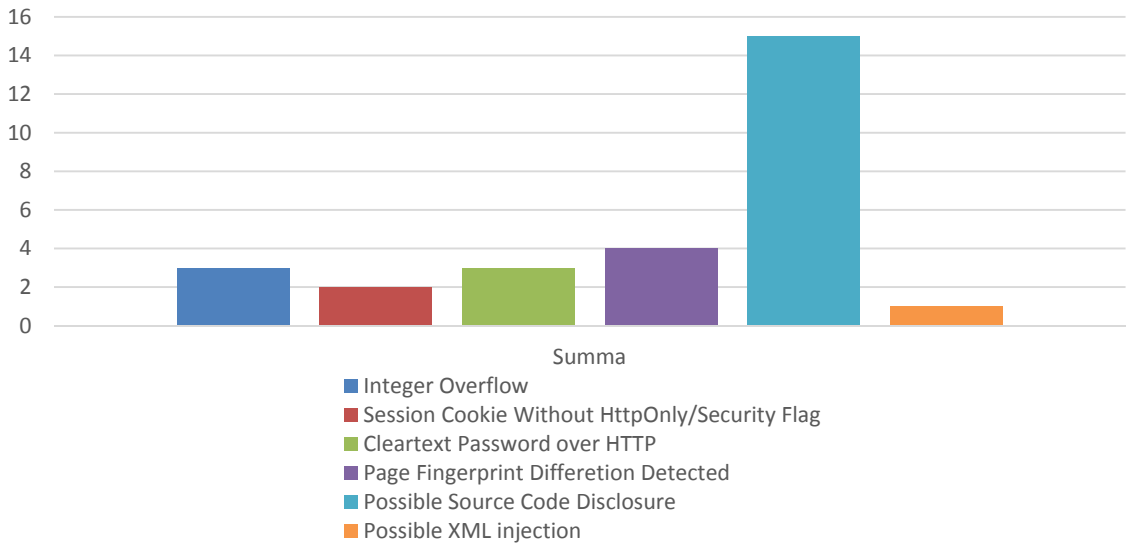
Vega-haavoittuvuusskannerilla on testattu kaikki seitsemän sovellusta ja löytyneet haavoittuvuudet ja niiden tasot on esitelty liitteessä 1. Haavoittuvuuksien tarkemmat kuvaukset on saatu skannerin antamista lisätiedoista. Kaaviossa 6.2. on esitetty löytyneet kriittisen- ja keskiluokan haavoittuvuudet sovelluskohtaisesti.

Taulukko 6.2 Vega-haavoittuvuusskannerin sovelluskohtaiset tulokset



Taulukosta huomataan, että jokainen testattu sovellus sisältää haavoittuvuuksia. Vega:n raportista saadaan lisätietoa, mistä sovelluksen tiedostosta tietty haavoittuvuus on löydetty. Mahdollisesti tunnistettu sosiaaliturvatunnus tai vakuutusnumero -haavoittuvuudet tarkistettiin ainoana haavoittuvuutena käsin skannauksen jälkeen ja ne havaittiin asiayhteyksistä vääriksi positiiviseksi. HTTP Trace Support Detected-haavoittuvuus johtuu testiympäristön Apache-palvelimen asetuksista ja Local Filesystem Paths Found-haavoittuvuus on seuraus tästä asetuksesta. PHP Error Detected-haavoittuvuus on tilanne, jossa sovellus antaa virheilmoituksen, josta on mahdollista lukea tietoja sovelluksen tilasta. Haavoittuvuus esiintyi selvästi Wordpress-sovelluksissa. Virheviestit voidaan kytkeä pois päältä PHP-asetuksista, eli se voidaan tulkita palvelimen virhekonfiguraatioksi. DVWA-sovelluksessa esiintyvät SQL- ja Shell-injektiot eivät esiinny muissa sovelluksissa, joten ne jätetään tarkastelun ulkopuolelle. Lisäksi istuntotunnisteeseen liittyvät haavoittuvuudet yhdistetään yhdeksi luokaksi ja haavoittuvuuksia ei jaotella sovelluskohtaisesti, niin saadaan taulukon 6.3. mukainen jaottelu.

Taulukko 6.3 Vega-haavoittuvuusskannerilla löytyneet huomioitavat haavoittuvuudet kaikissa sovelluksissa



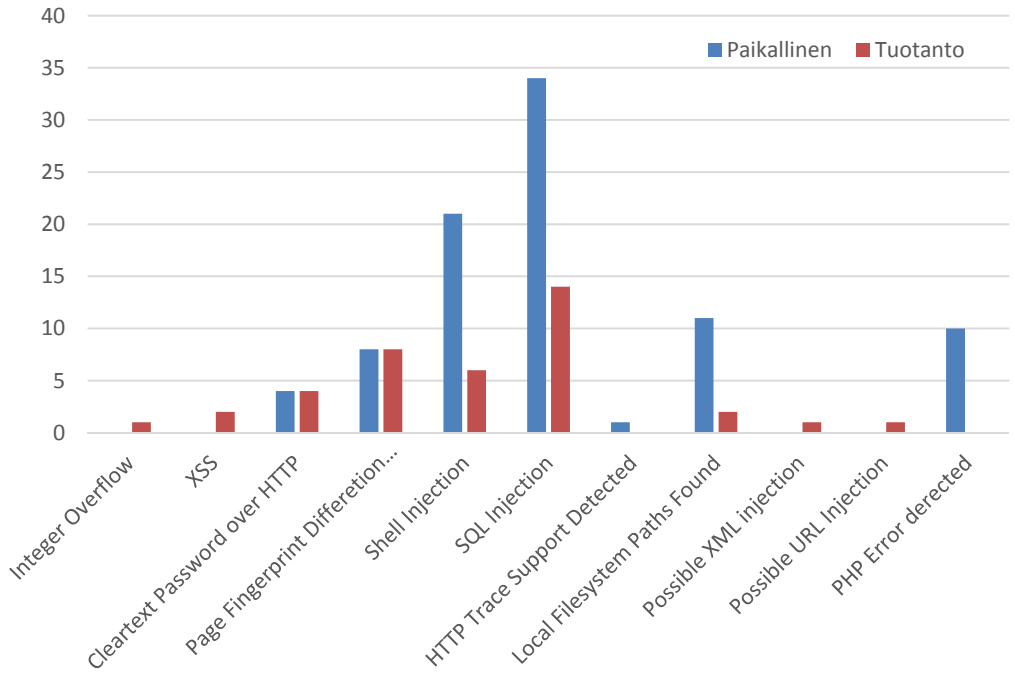
Löytyneistä haavoittuvuuksista neljä ylintä ovat vakavampia kriittisen tason haavoittuvuuksia. Haavoittuvuuksien esittely perustuu Vega-haavoittuvuusskannerin sisäiseen haavoittuvuusraporttiin. Integer Overflow-haavoittuvuus on tilannetta, jossa kokonaisluku-muuttujan arvo ylittää maksimaalisen asetusarvonsa. Haavoittuvuus voi aiheuttaa esimerkiksi puskurin ylivuodon, joka mahdollisesti johtaa arkaluotoisen tiedon vuotamiseen. Sovelluskehittäjän on suositeltavaa testata haavoittuvuus raportin pohjalta. Istuntotunnisteen käsittely ilman tietoturvatoimia mahdollistaa niiden muokkaamisen asiakasohjelmassa ja tunnisteiden salakuuntelun verkossa. Haavoittuvuus voidaan korjata sovel-

luksen ohjelmakoodissa, mutta tällaisenaan sovellus on vaarassa tunnistetietojen kaappaukselle ja muokkaukselle. Samoin salasanan lähettäminen ilman HTTP-yhteyden salaamista mahdollistaa tunnistetietojen kaappauksen verkkoliikennettä kuuntelemalla. Haavoittuvuus voidaan korjata ottamalla SSL-sertifikaatti käyttöön tai salaamalla liikenne jollain vaihtoehtoisella tavalla. Page Fingerprint Differention Detected-haavoittuvuus on tilanne, jossa selain lähettää sovellukselle resurssipyynnön, mutta resurssi palautetaan eri lähteestä, tai sen lähetys sisältää viitteitä paikalliseen tiedostoon. Tiedon avulla hyökkääjä voi löytää tiedostoja, joihin hänen ei pitäisi päästä käsiksi. Haavoittuvuus johtuu oletettavasti verkkopalvelimen tai sovelluksen asetuksissa ja voidaan välttää palvelimelle lisättävällä polkujenhallintaan suunnitellulla komentosarjakielen kirjastolla.

Keskitason haavoittuvuuksia ovat mahdollinen lähdekoodin paljastuminen ja mahdollinen XML-injektio. Mahdollinen lähdekoodin paljastuminen tarkoittaa haavoittuvuutta, jossa Vega on sovelluksen kartoittamisen yhteydessä löytänyt oletettavasti sovelluksen lähdekoodia. Paljastunutta lähdekoodia voidaan käyttää sovelluksen analysointiin ja haavoittuvuuden seuraukset voivat olla järjestelmän sisältämien vakavampien haavoittuvuuksien löytäminen. XML-injektio on tilanne, jossa ei-tarkistettua tietoa käytetään XML-tiedoston luomiseen. Vaikutus voi olla tiedon luottamuksellisuuden menettäminen. Keskitason haavoittuvuuksista huomataan, että ne eivät suoraan altista sovellusta tietoturvahyökkäykselle, vaan mahdollistavat vakavampien hyökkäyksien suorittamisen.

DVWA-sovellus asennettiin hetkellisesti toimimaan myös tuotantoympäristöön. Näin saadaan vertailevia tuloksia miten tuotantoympäristön palvelin- ja palomuuriasetukset vaikuttavat testauksen tuloksiin. Tulokset testistä on esitetty taulukossa 6.4.

Taulukko 6.4 **Vega-haavoittuvuusskannerilla löytyneet huomioitavat testi- ja tuotantoympäristössä**

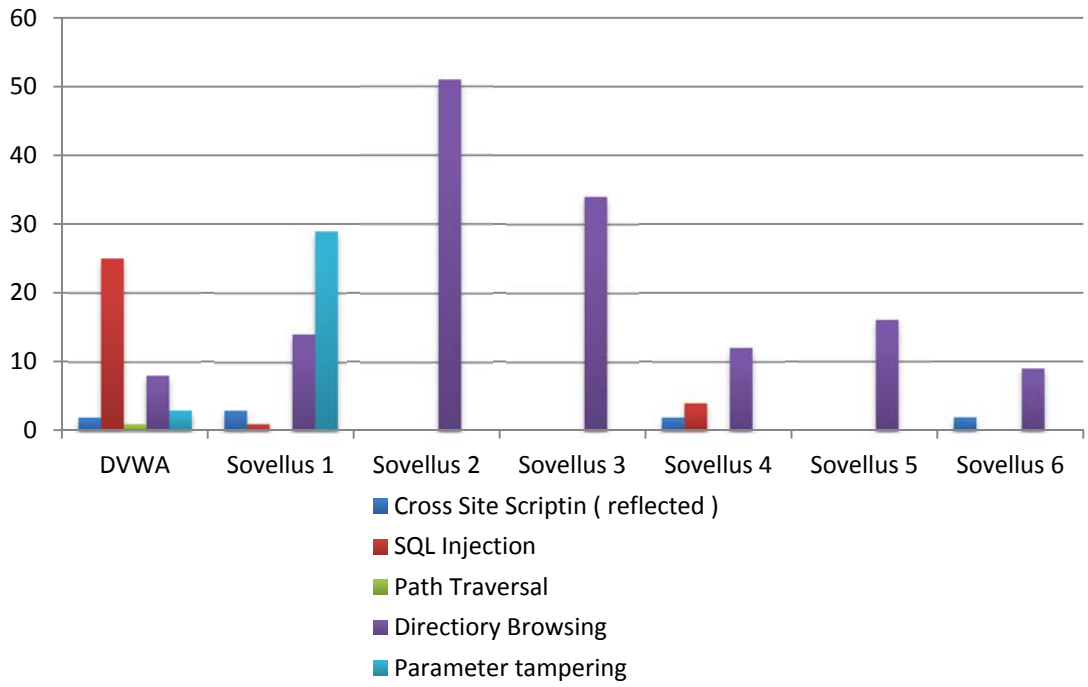


Ennen testauksen aloittamista tuotantoympäristössä oletus oli, että palvelun palomuuri estäisi skannauksen. Näin ei käynyt, ja skannaus voitiin suorittaa samoin kuin testausympäristössä. Kun paikallisen ja tuotantoympäristön tuloksia vertaillaan, huomataan, että haavoittuvuudet toistuvat myös tuotantoympäristössä. Haavoittuvuuksia on määrällisesti vähemmän, mutta eri haavoittuvuusluokkia on useampia. Vain kahdessa haavoittuvuudessa yhdestätoista saadaan sama tulos. Yhden sovelluksen perusteella ei voida tehdä johtopäätöksiä, mutta testiympäristöä voidaan pitää kohtuullisesti toimivana ja suuntaa-antavana mittauskohteena. Luotettavia tuloksia varten haavoittuvuusskannaus on suositeltavaa tehdä aidossa tuotantoympäristössä.

6.1.3 OWASP ZAP

OWASP ZAP-haavoittuvuusskannerilla testattiin kaikki seitsemän sovellusta ja tulokset kriittisistä ja keskitason haavoittuvuuksista on esitetty taulukossa 6.5. Kaikki saadut tulokset on esitetty liitteessä 2.

Taulukko 6.5 OWASP ZAP-haavoittuvuusskannerilla löytyneet haavoittuvuudet



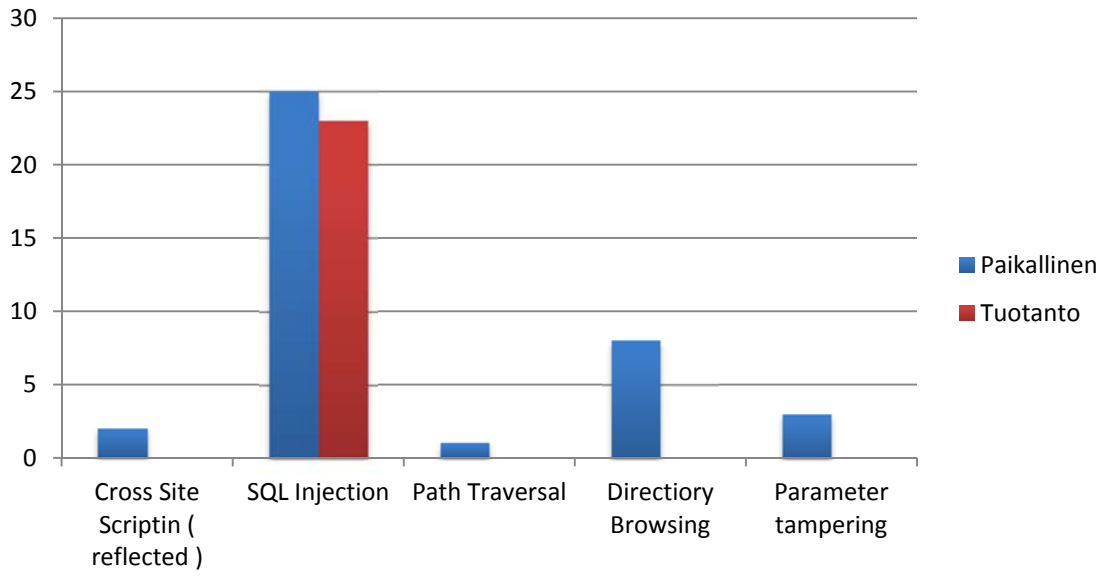
Erityyppisiä kriittisen- ja keskitason haavoittuvuuksia löytyi viittä eri tyyppiä. Kuten Vegassa, myös OWASP ZAP-löysi DVWA-sovelluksesta eniten eri haavoittuvuuksia ja muiden sovellusten kesken haavoittuvuudet jakautuivat epätasaisesti. Toisin kuin Vegassa, Wordpress-sovelluksia ei voida tunnistaa heti tuloksista näkyvinä piikkeinä.

Kolme ensimmäistä haavoittuvuutta ovat luokiteltu kriittisen tason uhkiksi. Ensimmäinen haavoittuvuus, eli Cross Site Scriptin on esitelty kappaleessa neljä. Toisin kuin Vega, OWASP ZAP löysi näitä haavoittuvuuksia kolmesta eri sovelluksesta. Kun

haavoittuvuuksia tutkittiin tarkemmin, huomattiin että DVWA-sovelluksessa haavoittuvuus oli linkitetty oikeaan kohdetiedostoon ja oli oikein tunnistettu. Muissa sovelluksissa jatkotoimenpiteitä ei tehty väärien positiivisten tarkistamiseksi, vaan haavoittuvuudet mainitaan asiakkaille toimitettavassa raportissa. Myös SQL-injektio löydettiin DVWA-sovelluksesta, sekä yhdestä testisovelluksesta. Samoin tämä täytyisi testata sovelluskohtaisesti ja lopullinen testaus jätetään sovelluksen ylläpitäjän tarkistettavaksi. Path Traversal -haavoittuvuus on tilanne, jossa verkkosovellus palauttaa verkkoresurssin osoitteen käyttäjän antamien syötteiden perusteella. Jos näitä tietoja ei validoida oikein, mahdollistaa tämä piilotettujen resurssin paljastumisen. Keskitason haavoittuvuuksia OWASP ZAP-sovellus tunnisti kahta eri tyyppiä. Directory Browsing -haavoittuvuus on tilanne, jossa sovelluksen tiedostorakenne paljastuu ei-tarkoitetulla tavalla ja tämä mahdollistaa piilotettujen tiedostojen paljastumisen. Parameter Tampering -haavoittuvuus perustuu verkkosovelluksen koskemattomiksi tarkoitettujen arvojen muokkaamiseen. Näitä voivat olla verkkosovelluksen piilotetut kentät, istuntotunniste tai sovelluksen sisältämät URL-osoitteet, joita ei ole tarkoitus esittää käyttäjälle. Kaikkia tällaisia kenttiä ei voida piilottaa hyökkääjältä, mutta haavoittuvuudelta voidaan suojautua validoimalla kenttien arvot ja niihin kohdistuvat muutokset [49].

Tuotantoympäristöön asetettu DWA-sovellus testattiin mys OWASP ZAP-haavoittuvuusskannerilla. Tulokset on esitetty taulukossa 6.6.

Taulukko 6.6 OWASP ZAP vertailu tuotanto ja testausympäristön välillä

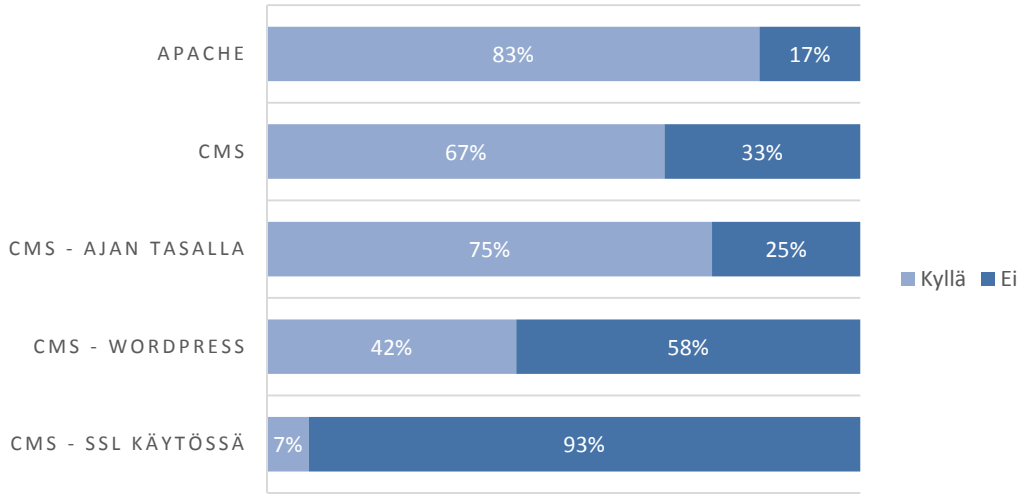


Taulukosta huomataan, että SQL_injektio oli ainoa molemmissa ympäristöissä esiintyvä haavoittuvuus. Tulos hieman yllätti, koska Vega-skannerilla löydettiin enemmän haavoittuvuuksia tuotantoympäristöstä kuin paikallisesta testausympäristöstä. SQL-haavoittuvuuksien osalta tulos on hyvin vastaava molemmissa ympäristöissä, joten haavoittuvuusskannerin voidaan olettaa toimineen oikein molemmissa mittauksissa.

6.1.4 Verkkopohjainen työkalu

Laajemman kokonaiskuvan saamiseksi kaikki dHosting-palvelussa olevat verkkosivut testattiin lisäksi verkossa toimivalla Sucuri-sovelluksella. Vaikka verkkopohjaisen sovellus ei testaa haavoittuvuuksien, saadaan siitä koko palvelua kattavaa asiakastietoa. Aikaisemman testin pohjalta todettiin, että Kalissa valmiiksi asennetut sovellukset eivät tarjoa tarpeeksi laadukasta tunnistamista ja tästä syystä päädyttiin käyttämään verkkopohjaista sovellusta. Taulukossa 6.7 on kuvatta tuloksia koskien koko palvelua ja siinä toimivia verkko-osoitteita. Verkko-osoitteet jotka osoittavat toiselle verkko-osoitteelle tai eivät sisällä ollenkaan sivuja, ei huomioida tuloksissa.

Taulukko 6.7 Verkkopohjaiseilla sovelluksilla mitattua tietoa dHsotign palvelun rakenteesta



Kuvasta huomataan, että 83 prosenttia palvelun asiakkaista sijaitsee Apache-verkkopalvelimella ja loput IIS-verkkopalvelimella. Palvelun asiakkaista 67 prosentilla on käytössä jokin sisällönhallintajärjestelmä. Löytyneitä julkaisujärjestelmiä olivat WordPress, Magento, MODX, TSI24, Clover Shop Ultrapro ja iWeb, joista 42 prosenttia oli Wordpress-julkaisujärjestelmiä. Tuloksissa julkaisujärjestelmien määrä oli hiemana suurempi kuin taulukon 2.1. tuloksissa, mutta WordPress-sovellusten määrä vähäisempi. Verkkopohjaisista työkaluista Sucuri tutki lisäksi löytyneen julkaisujärjestelmän versionumeron ja antaa varoituksen, jos sovellus ei ollut ajan tasalla tai vastannut uusin suositeltua julkaisua. 75 prosenttia sovelluksista oli ajan tasalla ja tältä osin tilanne oli hyvä. Huolestuttava tieto on, että vain seitsemällä prosentilla verkkosovelluksista on SSL-salaus käytössä. Kaikki sovellukset oletettavasti välittävät tietoa sovelluksen ja selaimen välillä ja suojaamattoman tiedon kaappaaminen on hyvin helppoa verkon solmukohdissa. SSL-sertifikaatin käyttöä ja hankkimista täytyy jatkossa helpottaa tarjoamalla sitä aktiivisesti asiakkaille. Samoin asian tärkeydestä täytyy tiedottaa palvelun myynnille ja lisätä tietoa verkkosivuille.

7 JOHTOPÄÄTÖKSET JA KORJAUSEHDOTUKSET

Diplomityön tavoite oli saavuttaa tarvittava ymmärrys verkkohotellipalvelun ylläpitämiseen liittyviin uhkiin ja pohtia asiakkaan ja palveluntarjoajan vastuita. Alkuperäisen palvelun tuottamiseen liittyvien komponenttien testauksen muuttaminen verkkosovelluksien haavoittuvuuksien tutkimiseen oli järkevää ja näin saatiin aikaan julkaistavia tuloksia, ilman että työtä tarvitsee julkaista salaisena. Ala on todella laaja ja ydinasioiden saaminen yhden diplomityön piiriin vaati haluttujen tietojen ja testattavan kohteen tarkkaa määrittelyä. Muuten vaarana on liian laajan kokonaisuuden käsittely. Monien tässä työssä esitettyjen yksittäisten asioiden, kuten haavoittuvuusskannerien, verkkosovellusten rakenteiden tai itse haavoittuvuuksien tutkimisesta olisi riittänyt materiaalia laajemmankin tutkimuksen tekemiseen. Henkilökohtaiset ja yritykseltä tulleet tavoitteet saatiin täytettyä ja palvelusta löydettiin parannusta kaipaavia osa-alueita. Samoin työn yhteydessä tehty asiakashaastattelut antoivat lisää tietämystä itse palvelunlaadusta ja kehitystarpeista.

Kappaleessa 4.2. esitetty väite, että 99 prosenttia sovelluksista sisältää haavoittuvuuksia, vaikutti ennen sovellusten testausta liioittelulta. Tässä työssä testatut sovellukset sisälsivät kaikki haavoittuvuuksia. Mahdollistaako jokin haavoittuvuus onnistuneen hyökkäyksen suorittamisen jää sovelluskehittäjien pohdittavaksi. Kaikki haavoittuvuudet ja uhat eivät johtuneet asiakkaiden tekemistä virheistä, vaan osa johtui verkkopalvelimen asetuksista. Lisäksi osa oli haavoittuvuusskannerin mielestä huonoa sovellussuunnittelusta ja eroja löytyi myös haavoittuvuusskannerien välillä. Määritelläänkö jokin tällainen ominaisuus haavoittuvuudeksi on tapauskohtaista ja jää sovelluskehittäjät voivat ottaa tietoisia riskejä parantaakseen sovellusten käytettävyyttä.

Haavoittuvuusskannerien käyttö oli todella helppoa ja on ymmärrettävää, että rikolliset automatisoivat niitä kohdesovellusten kartoittamiseen. Yksinkertaisimmillaan ainoa tarvittava syöte on kohteen verkko-osoite ja ohjelma tulostaa valmiin raportin haavoittuvuuksista. Testauksen aikana OWASP ZAP-skanneria testattiin lisäksi tuotantoympäristöön testaustarkoituksessa asennettuun WordPress-sovellukseen. Skanneri testasi generoiduilla syötteillä sovelluksen syötekenttiä, joista yksi oli yhteydenottolomake ylläpidolle. Sovelluksessa syötteiden jättämiseen ei ollut asetettu mitään rajoitusta ja sovellus lähetti 2400 sähköpostia ylläpidon sähköpostiin. Sama määrä roskapostia asiakkaalle olisi todennäköisesti johtanut yhteydenottoon. Tämä on hyvä esimerkki, miksi testaus aina täytyy sopia asiakkaan ja asiakkaan ylläpidon kanssa. Samalla täytyy kysyä pitäisikö PHP-asetuksista määritellä kaikille asiakkaille rajat sähköpostien lähettämiseen, vai voidaanko luottaa että asiakkaat itse hoitavat itse asetusten säätämisen hallintapaneelista. Kyseessä on tasapaino käytettävyyden ja tietoturvan välillä. Parempi lähestyminen voisi olla tiukempien rajojen asettaminen, joita asiakas voisi myöhemmin avata nähdessään tälle tarvetta.

Määrällisesti suurin osa haavoittuvuuksista oli matalan tason haavoittuvuuksia, ja ainoastaan yksi kohdesovellus ei sisältänyt kriittisen-tason haavoittuvuuksia. DVWA-testisovellus sai eniten eri haavoittuvuuksia molemmilla skannereilla. Haavoittuvuusskannerit löysivät haavoittuvuuksia niin paikallisesta kuin tuotantoympäristössä, ja erot ympäristöjen välillä olivat pienempiä, kuin ennen mittauksia oletettiin. Voidaan todeta, että testausympäristön pystyttäminen onnistui työssä halutulla tavalla. Skannerien osalta ongelma oli, että tulosten haavoittuvuusluokat eivät olleet keskenään verrattavissa.

Työn tuloksista tehdään myös lyhyet raportit testeihin osallistuneille asiakkaille. OWASP ZAP tarjoaa mahdollisuuden raportin tulostamiseen useissa eri tiedostomuodoissa ja tämä helpottaa raportoin luontia asiakkaalle. OWASP ZAP-haavoittuvuusskanneri ei testeissä löytänyt yhtä montaa haavoittuvuutta kuin Vega, mutta sen paremmin standardeja noudattava raportointi ja aktiivisempi kehitystyö tekevät siitä mielenkiintoisen ratkaisun tietoturvaskannauksien suorittamiseen myös asiakkaille maksullisena palveluna. SQL-injektio oli ainoa CWE-luokituksen mukainen haavoittuvuus molemmissa sovelluksissa. Ero löytyneissä haavoittuvuuksissa tuotanto ja testausympäristön eri skannereilla oli kohtuullisen pieni. Testausympäristössä tulokset olivat Vegassa 34 ja OWASP ZAP:issa 25 kappaletta. Ero skannerien kesken poikkeavista tuloksissa voi selittyä sillä, minkä tilanteen ohjelma tulkitsee haavoittuvuudeksi. Muuten haavoittuvuusskannerien tulokset eivät ole vertailukelpoisia keskenään. Jos verkkohotellipalveluun halutaan ottaa käyttöön haavoittuvuuksiin liittyviä palveluja, helpottaa tulosten tulkintaa, jos raportti annetaan esimerkiksi viitaten juuri CWE-luokkiin.

Koska dHosting palveluntarjoajana ei halua ottaa kantaa asiakkaiden verkkohotellien sisältöön, eikä tarjota verkkosovellusten ohjelmointiin tai ylläpitoon liittyviä palveluja, voidaan tämän työn perusteella sanoa, että verkkosovellusten tietoturvatestaus on järkevää jättää asiakkaiden vastuulle. Verkkosivuille voidaan lisätä ohjeet OWASP ZAP-sovelluksen käyttöön, mutta työstä saatujen tulosten ja käyttökokemusten perusteella palvelua ei ole järkevää tuotteistaa. Asiakkaiden sovelluksia ei tunneta tarpeeksi hyvin ja vastuuta asiakassovelluksille aiheutuneista ongelmista ei haluta ottaa. Lisäksi virheiden lopullinen korjaaminen jää silti lopulta asiakkaan vastuulle. Vaikka teknillisesti vaikuttaa, että tarve on olemassa, vaatisi tuotteistaminen laajemman asiakastutkimuksen kaupalliselta tarpeesta.

Palveluun on asennettu erillinen palomuuri, mutta se ei tunnistanut verkkoliikenteen seasta haavoittuvuusskannerien liikennettä. Palomuureissa on tunkeilijan havaitsemisjärjestelmiä (Intrusion Sedection Systems, IDS). IDS-järjestelmä analysoi verkkoliikennettä ja hälyttää poikkeavasta liikenteestä sekä ennalta tunnetuista vihamielisistä IP-osoitteista. Palomuurit voivat olla jatkuvassa yhteydessä niiden valmistajan palvelimelle, josta ne päivittävät aina uusimpiin havaintoihin perustuvat tiedot. Järjestelmät hälyttävät poikkeuksista järjestelmien ylläpitäjille, tai palomuuri voi tehdä päätöksen liikenteen estämisestä automaattisesti. Liikennettä analysoivia ja hallinnoivia palomuureja kutsutaan murronestojärjestelmiksi (Intrusion Prevention Systems, IPS) [75]. Hälytyksen aiheuttavia tilanteita ovat juuri haavoittuvuusskannauksen tapaiset, yhdestä osoitteesta hetkellisesti tulevat liikennepiikit. Taulukossa 6.1 on esitetty yhden testauksessa käytetyn

verkko-osoitteen liikenne kevään 2014 ajalta. Verkko-osoitteessa ei toimi mikään julkinen verkkosovellus, vaan sitä käytetään dHosting-palvelun kehittämiseen ja testaukseen. Verkko-osoite sijaitsee kuitenkin samassa tuotantoympäristössä, kuin palvelun asiakkaat.

Taulukko 6.1 Yhden Testaukseen käytetyn verkko-osoitteen liikenne kevään 2014 aikana

Mittari/KK	Tammi	Helmi	Maalis	Huhti	Touko
Eri IP osoitteita	33	94	292	8	2
Vierailujen määrä	50	788	551	15	3
Suoritetut pyynnöt	4861	2066	1302	33	22187

Taulukosta huomataan, että alkuvuoden aikana verkko-osoitteessa on ollut normaalia, vaikka kohtuullisen vähäistä testausliikennettä. Toukokuussa suoritetut skannaukset verkko-osoitteessa sijaitseville kahdelle kohdesovellukselle kahdella eri haavoittuvuusskannerilla nostivat suoritettujen resurssipyyntöjen määrän yli 22 000 kappaleeseen. Tämä on noin kymmenen kertaa enemmän kuin kuukauden normaali liikenne ja kaikki pyynnöt tulivat yhdestä IP-osoitteesta. Tämän kaltaiset kävijämäärässä tapahtuvat muutokset todennäköisesti hukkuisivat suosittujen sivustojen normaalin liikenteen sekaan, mutta juuri lyhyen ajanjakson sisällä yhdestä IP-osoitteesta tulevat toistuvat pyynnöt voisivat aiheuttaa hälytyksen. Hyökkääjä voi välttää tällaisia piikkejä jakamalla skannauksen pidemmällä aikavälille sekä usealle välittäjäpalvelimelle ja näin ohittaa esimerkiksi liikennemäärään perustuvat rajoitukset. Näiden tulosten perusteella olisi järkevää hankkia palvelun palomuuriin verkkohotellipalvelun verkkoliikennettä analysoiva ja eri hyökkäyksiä tunnistava lisäominaisuus. Tietoturvaa lisäävien ominaisuuksien käyttöönotto onnistuu yksinkertaisimmillaan palomuuriin ostettavan lisälisenssin aktivoinnilla.

Toinen havainto diplomityön tuloksista oli, että vain 7 prosenttia verkkosovelluksia ylläpitävistä asiakkaista salasi HTTP-liikenteensä. Kuten luvussa kaksi todetaan, HTTP-liikenne ei ole itsessään tietoturvallinen protokolla. dHosting palvelu tarjoaa mahdollisuuden SSL-lisenssin käyttöön ja asiasta on lisättävä tietoa niin asiakkaiden kuin myyjien osalta. Myös palvelun verkkosivuille on suositeltavaa laittaa kattavampaa tietoa itse tietoturvaongelmista, sekä sertifikaatin käyttöönotosta. On mahdollista, että verkkosovelluksen sisälle on rakennettu oma salausmetodi, mutta koko liikenteen salaaminen on suositellumpi ratkaisu.

Kuten tietoturvatestauksen aloittamista käsittelevässä kappaleessa todetaan, täytyy testaukselle määritellä myös loppu. Nämä tulokset ovat hyvä kohta rajata tutkimus. Mielenkiintoista olisi lisäksi tutkia, miten asiakkaat asentavat verkkosovelluksensa, kuinka he ylläpitävät niitä ja kuinka paljon asiakas luottaa, että verkkohotellipalveluntarjoaja huolehtii sovellusten tietoturvasta. Julkaisujärjestelmien yleistyminen ja verkkohotellipalvelun tarjoamat valmiit asennuspaketit mahdollistavat verkkosovellusten asentamisen ilman minkäänlaista ohjelmointiosaamista tai alan tuntemusta. Olisi mielenkiintoista selvittää kuinka paljon asiakkaat miettivät sivujen jatkuvaa päivittämistä ja tietoturvaa jatkuvana prosessina. Jos sivujen ohjelmointi ostetaan ulkopuoliselta yritykseltä,

nähdäänkö hankinta kertaluontoisena ostona, vai myydäänkö sivut jatkuvana palveluna, johon kuuluu myös tietoturvan ylläpito.

8 LÄHTEET

1. Webhotellivertailu, Suomen kattavin webhotellivertailu. [WWW] Saatavissa: www.webhotellivertailu.fi, [Viitattu 9.4.2014]

2. Tarja Huovinen (2003), Palvelun laatuun ja sen johtamiseen liittyvät ongelmat : Esimerkkinä kylpylät, Yrittäjyyden Pro Gradu Tutkielma. Jyväskylän yliopiston taloustieteiden tiedekunta, Sivut: 8-9

3. Paulus Mikkola (2012), Verkkosovelluksen määritelmä, etusivu, kappale 2, [WWW] Saatavissa: www.verkkosovellukset.com [Viitattu 10.4.2014]

4. Hugo Haas ,W3C & Allen Brown, Microsoft (2004) W3C Working Group Note 11 February 2004, Kappale kaksi, määritelmät: Web service, [WWW] Saatavissa: http://www.w3.org/TR/2004/NOTE-ws-gloss-20040211 [Viitattu 3.5.2014]

5. Michal Zalewski (2012), THE TANGLED WEB, A Guide to Securing Modern Web Applications, No Starch Press, San Francisco, 1. painos. ISBN-10: 1-59327-388-6, Sivut: 9-13, 17-18, 42-43, 47, 60-61, 76. 95-96, 97-101, 109-110

6. Mike Shema (2012), Hacking Web Apps, Detecting and Preventing Web Application Security Problems, Syngress, 225 Wyman Street, Waltham, MA 02451, USA, ISBN: 978-1-59749-951-4, Sivut: xvi, 1-3, 3-6, 72, 142, 143-144, 109-11, 117-121, 141-142, 144, 146, 165, 201-202

7. R. Fielding, UC Irvine, J. Gettys (1999), Hypertext Transfer Protocol -- HTTP/1.1, Network Working Group, RFC: 2616, Kappaleet 1.1 & 1.4, [WWW] Saatavissa: tools.ietf.org/html/rfc2616 [Viitattu 14.4.2014]

8. Joel Scamray, Mike Shema (2002), Hacking Exposed Web Applications, McGraw-Hill/Osborne, 2600 Tenth Street, Berkeley, California 94710, U.S.A. ISBN 0-07-222438-X, Sivut: 5, 14, 16-17

9. OWASP, Open Web Application Security Project (2013), OWASP Top 10 – 2013, The Ten Most Critical Web Application Security Risks, The OWASP Foundation, Creative Commons (CC), Sivut: 6-8, 10-12, 22, 38, [WWW] Ilmainen versio ladattavissa: www.owasp.org [Viitattua 7.9.2014]

10. E. Rescorla, Network Working Group (2000), HTTP Over TLS, RFC 2818, Kappale 2, [WWW] Saatavissa: www.ietf.org/rfc/rfc2818.txt, [Viitattu 30.12.2013]

11 Dierks & Rescorla, Network Working Group (2008), The Transport Layer Security (TLS) Protocol Version 1.2, [WWW] Saatavussa: http://tools.ietf.org/html/rfc5246, [Viitattu 30.12.2013]

12. Tubewar (2010), How SSL works tutorial - with HTTPS example, [WWW] Saatavussa: www.youtube.com/watch?v=iQsKdtjwtYI, [Viitattu 31.12.2013]

13. Q-Success, W3Techs - World Wide Web Technology Surveys, [WWW] Saatavissa: w3techs.com, [Viitattu 10.4.2014]

14. Netcraft (Helmikuu 2014), February 2014 Web Server Survey, [WWW] Saatavissa: news.netcraft.com/archives/2014/02/03/february-2014-web-server-survey.html#more-13987, [Viitattu 22.4.2014]

15. Peter Pollock (2013), Web Hosting For Dummies®, John Wiley & Sons, Inc. 111 River Street, Hoboken, NJ 07030-5774, www.wiley.com, ISBN 978-1-118-54042-8, Sivut: 24-27, 30-31, 39-54, 58-60,

16. Parallels IP Holdings GmbH (1999 - 2014), Administrator's Guide, Parallels Plesk Panel 11.5, Web Servers [WWW] Saatavissa: download1.parallels.com/Plesk/PP11/11.5/Doc/en-US/online/plesk-administrator-guide/index.htm?fileName=71932.htm, [Viitattu 7.1.2014]

17. Microsoft, Developer Network, Chapter 21: Designing Web Applications, [WWW] Saatavissa: msdn.microsoft.com/en-us/library/ee658099.aspx, [Viitattu 14.4.2014]

18. Jari-Pekka Vuotilainen (2011), Vuorovaikutteisten web-sovellusten kehittäminen, Diplomityö, Tampereen teknillinen yliopisto, Tietotekniikan koulutusohjelma, Sivut: 1-2

19. Amit Agarwal, Digital Inspiration (2009), Web 3.0 Concepts Explained in Plain English, [WWW] Saatavissa: www.labnol.org/internet/web-3-concepts-explained/8908, [Viitattu 7.4.2014]

20. O.S Tezer, DigitalOcean (2014), Understanding SQL And NoSQL Databases And Different Database Models, [WWW] Saatavissa: www.digitalocean.com/community/articles/understanding-sql-and-nosql-databases-and-different-database-models, [Viitattu 24.4.2014]

21. J O'Dell, Venture Beat, Kesäkuu (2013), 19 percent of the web runs on WordPress, [WWW] Saatavissa: www.venturebeat.com/2013/07/27/19-percent-of-the-web-runs-on-wordpress, [Viitattu 10.4.2014]

22. Versa Studio LLC, Madison, Wisconsin, Why use a Content Management System for your web site? [WWW] Saatavissa: www.versastudio.com/articles/why-use-a-content-management-system-for-your-web-site, [Viitattu 10.4.2014]

23.Jeremiah Shoaf, Type & Grids (2013), Goodbye WordPress: 2014 Will Be the Year of the Flat-File CMS, [WWW] Saatavissa: www.typeandgrids.com/blog/goodbye-wordpress-2014-will-be-the-year-of-flat-file-cmses, [Viitattu 10.4.2014]

24. Webopedia, Quinstreet Enterprise, Verkkosanakirja: Virtual Host, [WWW] Saatavissa: www.webopedia.com/TERM/V/virtual_host.html, [Viitattu 7.1.2014]

25. Parallels IP Holdings GmbH (1999 - 2014), Administrator's Guide, Parallels Plesk Panel 11.5, Managing Customers and Subscriptions, [WWW] Saatavissa: download1.parallels.com/Plesk/PP11/11.5/Doc/en-US/online/plesk-administrator-guide/index.htm?fileName=70729.htm, [Viitattu 22.4.2014]

26. Joe Jensen, Networkbits (2008), What Type of Web Hosting Is Best For You? [WWW] Saatavissa: www.networkbits.net/web/web-hosting-types, [Viitattu 16.12.2013]

27. The Apache Software Foundation, Name-based Virtual Host Support, [WWW] Saatavissa: http://httpd.apache.org/docs/2.2/vhosts/name-based.html, [Viitattu 9.1.2014]

28. Parallels IP Holdings GmbH (1999 - 2014), Administrator's Guide, Parallels Plesk Panel 11.5, Plesk API-RPC Guide, [WWW] Saatavissa: download1.parallels.com/Plesk/PP10/10.1.1/Doc/en-US/online/plesk-api-rpc-guide/index.html, [Viitattu 16.12. Plesk API]

29. Parallels IP Holdings GmbH (1999 - 2014), Administrator's Guide, Parallels Plesk Panel 11.5, About Parallels Plesk Panel, [WWW] Saatavissa: download1.parallels.com/Plesk/PP11/11.5/Doc/en-US/online/plesk-administrator-guide/index.htm?fileName=about.htm, [Viitattu 9.1.2014]

30. Q-Success, Usage statistics and market share of PHP for websites, [WWW] Saatavissa: http://w3techs.com/technologies/details/pl-php/all/all, [Viitattu 9.4.2014]

31. Ohjelmointinurkka, YLEISTÄ PHP-OHJELMOINTIKIELESTÄ, [WWW] Saatavissa: http://ohjelmointi.medianurkka.com/?p=21, [Viitattu 9.4.2014]

32. Parallels IP Holdings GmbH (1999 - 2014), Administrator's Guide, Parallels Plesk Panel 11.5, PHP Configuration, [WWW] Saatavissa: http://download1.parallels.com/Plesk/PP11/11.5/Doc/en-US/online/plesk-administrator-guide/index.htm?fileName=70668.htm, [Viitattu 9.1.2014]

33. Abrahim Ikasud (2009), GETTING & MANAGING YOUR FIRST 1,000 CLIENTS IN WEB HOSTING, Ichthys Media LLC 39380 Civic Center Dr #307 Fremont, CA 94538, Sivut: 147-148

34. Matti Laakso (2010), PK-YRITYKSEN TIETOTURVASUUNNITELMAN LAATIMINEN, Opinnäytetyö, Turun Ammattikorkeakoulu, Sivut 7, 26, 16

35. Charles P. Pfleeger, Shari Lawrence Pfleeger (2006), Security in Computing, Neljäs painos, Prentice Hall, ISBN-10: 0-13-239077-9, Sivut: 5-10, 15, 29-37, 39-44

36. Irwing Lachow, CNAS, (2013), Active Cyber Defense, A Framework for Policymakers, Sivu 2, [WWW] Saatavissa: www.cnas.org/files/documents/publications/CNAS_ActiveCyberDefense_Lachow_0.pdf, [Viitattu 20.12.2013]

37. Jan Mickos, CGI, Tietoviikko (6.6.2013), Kävikö kyberrosvo kylässä? [WWW] Saatavissa: www.tietoviikko.fi/viisaat/cgi/kaviko+kyberrosvo+kylassa/a907261, [Viitattu 20.12.2013]

38. Valtionhallinnon tietoturvallisuuden johtoryhmä (2012), Teknisen ICT-ympäristön tietoturvataso-ohje, VM/1991/00.00.00/2012, Sivut 14-16, [WWW] Saatavissa: http://www.vm.fi/vm/fi/04_julkaisut_ja_asiakirjat/01_julkaisut/05_valtionhallinnon_tietoturvallisuus/20121122Teknis/ICT_taitto.pdf, [Viitattu 7.9.2014]

39. Michael Cobb, ComputerWeekly (2009), Cross-site scripting explained: How to prevent XSS attacks, [WWW] Saatavissa: www.computerweekly.com/tip/Cross-site-scripting-explained-How-to-prevent-XSS-attacks, [Viitattu 6.9.2014]

40. Mike Wasson, Microsoft (2012), Preventing Cross-Site Request Forgery (CSRF) Attacks, [WWW] Saatavissa: www.asp.net/web-api/overview/security/preventing-cross-site-request-forgery-(csrf)-attacks, [Viitattu 6.9.2014]

41. Vinnie Murdico, Software with Brains, Inc (2007), Bugs per lines of code, [WWW] Saatavissa: www.amartester.blogspot.fi/2007/04/bugs-per-lines-of-code.html, [Viitattu 7.9.2014]

42. Margaret Rouse, SearchSecurity (2012), Attack vector, [WWW]
Saatavissa: http://searchsecurity.techtarget.com/definition/attack-vector,
[Viitattu 5.12.2013]

43. Laihonen, Hannula, Helander (2013), Tietojohtaminen, Tampereen teknillinen yliopisto, Tiedonhallinnan ja logistiikan laitos, Juvenes Print, Tampere, ISBN 978-952-15-3057-9. Sivu 43

44. Jarmo Pihlajaniemi (2012), REST-pohjaisen web-rajapinnan kehittäminen, Metropolia Ammattikorkeakoulu, Tietotekniikan koulutusohjelman Insinöörityö, Sivut: 19-20

45. Jussi Kari (2010), Web 2.0 -palveluntarjoajan vastuu erityisesti käyttäjien tekemistä tekijänoikeuden loukkauksista, Opinnäytetyö, Turun Yliopiston Oikeustieteellinen tiedekunta. 2010, Sivut: 32,55

46. Brian Krebs (2013), Web Badness Knows No Bounds, [WWW] Saatavissa: krebsonsecurity.com/2013/06/web-badness-knows-no-bounds/, [Viitattu 3.2.2014]

47. Cantic, Application Vulnerability Trends Report (2014), Application Vulnerability Trends Report: 2014, Sivu 6, [WWW] Saatavissa: info.cenzic.com/rs/cenzic/images/Cenzic-Application-Vulnerability-Trends-Report-2013.pdf, [Viitattu 10.2.2014]

48. Positive Research (2011), Vulnerability Statistics for 2011, Sivu: 6, [WWW]
Saatavissa: www.ptsecurity.com/download/Vulnerability-Statistics-for-2011.pdf,
[Viitattu 10.2.2014]

49. MITRE Corporation, (2014), A Community-Developed Dictionary of Software Weakness types, Mahdollisuus hakea eri haavoittuvuuksia ID-numeron perusteella, [WWW] Saatavissa: cwe.mitre.org, [Viitattu 3.6.2014]

50. Jeremy Ferragamo, OWASP (2010), Injection Flaws, [WWW]
Saatavissa: www.owasp.org/index.php/Injection_Flaws, [Viitattu 31.3.2014]

51. KirstenS, OWASP (2010), SQL Injection, [WWW]
Saatavissa: www.owasp.org/index.php/SQL_Injection, [Viitattu 22.3.2014]

52. Patrick Thomas, BlindElephant Web Application Fingerprinter, Sovelluksen kotisivu, [WWW] Saatavissa: blindelephant.sourceforge.net, [Viitattu 6.9.2014]

53. Simon Bennetts, OWASP Zed Attack Proxy Project (2014), Sovelluksen kotisisu, [WWW] Saatavissa : www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project, [Viitattu 6.9.2014]

54. Andrew Muller, OWASP (2010), Testing for Session Management, [WWW] Saatavissa: Saatavissa: www.owasp.org/index.php/Testing_for_Session_Management, [Viitattu 1.4.2014]

55. Keir Thomas, PCWorld (2011), Password Reuse Is All Too Common, Research Shows, [WWW] Saatavissa: www.pcworld.com/article/219303/password_use_very_common_research_shows.html, [Viitattu 1.4.2014]

56. MITRE Corporation (2014), CWE-287: Improper Authentication, [WWW] Saatavissa: cwe.mitre.org/data/definitions/287.html, [Viitattu 1.4.2014]

57. Jane O'Connor, OWASP (2014), Test Role Definitions, [WWW] Saatavissa: https://www.owasp.org/index.php/Test_Role_Definitions_(OTG-IDENT-001), [Viitattu 1.4.2014]

58. Acutinex, Cross Site Scripting – XSS – The Underestimated Exploit, [WWW] Saatavissa: https://www.acunetix.com/websitesecurity/xss/, [Viitattu 3.2.2014]

59 Corey Nachreiner, Net Security (2014), Defending Against Drive-by Downloads, [WWW] Saatavissa: www.net-security.org/article.php?id=1946&p=1, [Viitattu 3.2.2014]

60. Neil DuPaul, VeraCode, Cross-Site Scripting (XSS) Tutorial: Learn About XSS Vulnerabilities, Injections and How to Prevent Attacks, [WWW] Saatavissa: www.veracode.com/security/xss, [Viitattu 10.2.2014]

61. Fraser Howard, SophosLabs, UK, (2012), Exploring the Blackhole Exploit Kit, [WWW] Saatavissa: sophosnews.files.wordpress.com/2012/03/blackhole_paper_mar2012.pdf, [Viitattu 4.2.2014]

62. Andrew Muller, OWASP (2014), Testing for CSRF, [WWW] Saatavissa: www.owasp.org/index.php/Testing_for_CSRF_(OWASP-SM-005), [Viitattu 1.4.2014]

63. Jeff Williams, OWASP, (2014),Vulnerability, [WWW] Saatavissa: www.owasp.org/index.php/Category:Vulnerability, [Viitattu 27.5.2014]

64. W3Schools, Web Services Tutorial, [WWW] Saatavissa: www.w3schools.com/webservices/default.asp, [Viitattu 6.9.2014]

65. Adrian Sanabria, Penetration Testing Execution Standard (2012), Pre-engagement, [WWW] Saatavissa: www.pentest-standard.org/index.php/Pre-engagement, [Viitattu 3.4.2014]

66. Nikolay Dimitrov, OWASP, (2014), Vulnerability Scanning Tools, [WWW] Saatavissa: www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools, [Viitattu 27.5.2014]

67 Oracle Corporation, (2012), Oracle VM VirtualBox, User Manual, Sivut: 12,93, [WWW] Saatavissa: download.virtualbox.org/virtualbox/UserManual.pdf, [Viitattu 17.2.2014]

68. Offensive Security Ltd. (2014), Kali Linux Ducumentation, [WWW] Saatavissa: docs.kali.org, [Viitattu 4.2.2014]

69. Plecost, Wordpress Fingerprinting Tool, Ohjelmaprojektin kotisivu, [WWW] Saatavissa: code.google.com/p/plecost, [Viitattu 13.3.2014]

70. Fakhreldeen Abbas Saeed, IJCSN (2014), Using WASSEC to Analysis and Evaluate Open Source Web Application Security Scanners Application Security Scanners, Sivut: 43-46, [WWW] Saatavissa: ijcsn.org/IJCSN-2014/3-2/Using-WASSEC-to-Analysis-and-Evaluate-Open-Source-Web-Application-Security-Scanners.pdf

71. Syed Balal Rumy, Rumy IT Tips (2013), How to use Vega Web Vulnerability Scanner in Kali Linux, [WWW] Saatavissa: www.rumyittips.com/how-to-use-vega-web-vulnerability-scanner-in-kali-linux, [Viitattu 12.3. 2014]

72. Subgraph, Using the Vega Scanner, Ohjeet Vega-haavoittuvuusskannerin käyttöön, [WWW] Saatavissa: https://subgraph.com/vega/documentation/Vega-Scanner/index.en.html, [Viitattu 20.5.2014]

73. NJ Ouchnm, Toolswatch (2013), 2013 Top Security Tools as Voted by Tools-Watch.org Readers, [WWW] Saatavissa: www.toolswatch.org/2013/12/2013-top-security-tools-as-voted-by-toolswatch-org-readers, [Viitattu 27.5.2014]

74. RandomStorm, Damn Vulnerable Web Application, Sovelluksen kotisivu, [WWW] Saatavissa: www.dvwa.co.uk, [Viitattu 6.9.2014]

75. Pietari Sarjakivi, Nixu, Hyökkäyksen havainnointi- ja estojärjestelmät (IPS & IDS), [WWW] Saatavissa: www.nixu.com/fi/palvelualueet/hy%C3%B6kk%C3%A4yksen-havainnointi-ja-estoj%C3%A4rjestelm%C3%A4t-ips-ids, [Viitattu 11.6.2014]

LIITTEET

Liite 1. Vega-Haavoittuvuusskannerin tulokset

Taso	Riski	DVWA	Sovellus 1	Sovellus 2	Sovellus 3	Sovellus 4	Sovellus 5	Sovellus 6	Summa
HIGH	Integer Overflow	0	0	0	3	0	0	0	3
HIGH	Session Cookie Without HttpOnly Flag	0	0	0	1	0	0	0	1
HIGH	Session Cookie Without Secure	0	0	0	1	0	0	0	1
HIGH	Cleartext Password over HTTP	4	0	1	0	1	0	1	7
HIGH	Page Fingerprint Differetion Detected	8	0	1	0	2	0	1	12
HIGH	Possible Security Number/Social Insurence Number Detected	0	2	4	0	4	0	2	12
HIGH	Shell Injection	21	0	0	0	0	0	0	21
HIGH	SQL Injection	34	0	0	0	0	0	0	34
Medium	HTTP Trace Support Detected	1	1	1	1	1	1	1	7
Medium	Local Filesystem Paths Found	11	61	67	3	61	0	65	268
Medium	Possible Source Code Disclosure	0	4	4	0	3	1	3	15
Medium	Possible XML injection	0	0	0	1	0	0	0	1
Medium	PHP Error derected	10	80	80	0	79	0	80	329
LOW	Directory Listing Detected	7	59	110	385	85	14	97	757
LOW	Email Addresses Found	1	3	3	50	4	1	3	65
LOW	Form Password Field witf Autocomplete Enabled	0	0	1	0	1	0	1	3
LOW	Internal Addresses Found	1	0	0	0	0	0	0	1
INFO	Blank Body Detected	1	100	111	8	112	2	109	443
INFO	Character Set Not Specified	22	84	99	548	83	1	102	939
INFO	Cookie http-Only Fag Not Set	1	0	1	1	2	0	1	6
INFO	HTTP Error Detected	0	0	0	3	0	0	0	3
INFO	Version Control String Fouind	0	0	0	1	0	0	0	1
INFO	News Feed Upload Detected	0	1	1	0	1	0	0	3
INFO	Form File Upload Detected	1	2	3	0	3	0	2	11
INFO	Possible AJAX code Detected	1	0	0	0	0	0	0	1
INFO	X-Frame Options Header Not Set	61	336	785	2363	523	62	662	4792
		DVWA	Sovellus 1	Sovellus 2	Sovellus 3	Sovellus 4	Sovellus 5	Sovellus 6	Summa

Liite 2. OWASP ZAP haavoittuvuusskannerin tulokset

Taso	Riski	DVWA	Sovellus 1	Sovellus 2	Sovellus 3	Sovellus 4	Sovellus 5	Sovellus 6	Summa	CWE ID
HIGH	Cross Site Scriptin (reflected)	2	3	0	0	2	0	2	9	79
HIGH	SQL Injection	25	1	0	0	4	0	0	30	89
HIGH	Path Traversal	1	0	0	0	0	0	0	1	22
Medium	Directiory Browsing	8	14	51	34	12	16	9	144	548
Medium	Parameter tampering	3	29	0	0	0	0	0	32	472
Low	Cross-domain JavaScript source file inclusion	1	6	85	0	0	1	0	93	0
Low	Private IP disclosure	70	16	220	2104	47	148	826	3431	200
Low	X-Content-Type-Options header missing	99	44	827	2329	92	281	1371	5043	0
Low	Cookie set without HttpOny flag	50	0	117	20	6	4	17	214	0
Low	Content type header missin	0	0	24	35	0	1	0	60	0
Low	Password autocomplete in browser	0	0	0	0	4	0	3	7	525
Info	X-Frame-Options header not set	93	36	51	2140	49	168	1227	3764	0